P. 1877/93

6 1993

informatyka

WYDAWNICTWO SIGMA NOT

Cena 25 000 zł

informatyka

nr 6 1993
Czerwiec
Rok wydania XXVIII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
Władysław KLEPACZ
(redaktor naczelny)
mgr inż. Jan RYŻKO
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Mazowiecka 12 telefon: 27-43-66 telefaks: 19-21-87 teleks: 814877

W numerze:

Strona

Rozproszone systemy operacyjne – *Zbigniew Weiss* 1
Badania ankietowe. Ocena poziomu zastosowań informatyki dla celów zarządzania w Polsce – *Stanisław Wrycza, Tomasz Plata-Przechlewski* 6
SR – obiektowo zorientowany język programowania rozproszonego – *Aleksander Laurentowski, Krzysztof Zieliński* 15
Algorytm rozpoznawania problemu szeregowania zadań – *Tadeusz Witkowski* 20
Nowe książki Wydawnictw Naukowo-Technicznych 27

W najbliższych numerach:

- Przemysław Pardyak i Krzysztof Zieliński charakteryzują język Emerald, umożliwiający obiektowo zorientowane programowanie współbieżne i rozproszone.
- Krzysztof Sacha opisuje projektowanie oprogramowania systemów wbudowanych tzn. systemów sterowania i dowodzenia.
- Tomasz Grzelak omawia systemy typu CAS, ze szczególnym uwzględnieniem funkcji wspomagania projektowania kompozycji graficznych w przemyśle reklamowym.
- Jan Zabrodzki przedstawia koncepcję nauczania grafiki komputerowej w ramach kierunku Informatyka.
- Michał Goliński relacjonuje wybrane aspekty zastosowań informatyki i telekomunikacji, podaje podstawowe definicje i trendy rozwojowe rynków elektronicznych.

Warunki prenumeraty na 1993 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573

Wpłaty na prenumeratę od marca br. przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 25 000 zł, ulgowa 18 750 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 75 000, półroczna 150 000, roczna 300 000
Ulgowa: kwartalna 56 250, półroczna 112 500, roczna 225 000.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 217/93

P. 1877/93
POLITECHNIKA ŚLĄSKA Biblioteka Główna Gliwice

ZBIGNIEW WEISS
Instytut Informatyki
Uniwersytet Warszawski

Rozproszone systemy operacyjne

System operacyjny to najbliższa sprzętowi warstwa oprogramowania komputerów. Rozwój systemów operacyjnych jest przez to do pewnego stopnia uzależniony od rozwoju techniki komputerowej. Tylko dzięki takim mechanizmom sprzętowym, jak: przerwania, instrukcje uprzywilejowane, stronicowanie, bity użycia i zapisu, ochrona pamięci można było zrealizować wielodostępne i wieloprogramowe systemy operacyjne z podziałem czasu i z pamięcią wirtualną.

Pojawienie się możliwości łączenia ze sobą odrębnych komputerów i przesyłania między nimi danych dało nowy bodziec do rozwijania systemów operacyjnych. Można właściwie powiedzieć, że był to przełom, który dla rozwoju techniki komputerowej może w przyszłości okazać się równie ważny, jak dla rozwoju ludzkości było niegdyś przejście od samowystarczalności do wymiany towarowej. Przestaliśmy bowiem wymagać, aby jeden komputer robił wszystko i pamiętał wszystko. Połączone ze sobą komputery mogą dzielić się informacją i zadaniami, zwiększając w ten sposób wydajność i efektywność przetwarzania.

Parafrazując cytat biblijny można by powiedzieć, że człowiek stworzył komputery na swój wzór i podobieństwo. Jest to oczywiście wytwór bardzo niedoskonały, niemniej istnieje wiele analogii między algorytmami stosowanymi w systemach operacyjnych, a naszym codziennym postępowaniem. Dotyczy to w szczególności komunikacji między komputerami. Sieć komputerowa działa w sposób bardzo podobny do sieci telegraficznej czy telefonicznej. Tak jak telefon jest dla człowieka narzędziem do wydawania poleceń i uzyskiwania informacji, tak samo z punktu widzenia pojedynczego komputera łącze z siecią jest jeszcze jednym urządzeniem wejścia-wyjścia, przez które komunikuje się on ze światem zewnętrznym. Jest to jednak bardzo specyficzne urządzenie wejścia-wyjścia, albowiem po drugiej stronie stoi nie człowiek, ale inny komputer.

Sieciowy system operacyjny

Scentralizowany system operacyjny zarządza takimi zasobami, jak: procesor, pamięć, dyski, drukarki, taśmy czy terminale. Nie może jednak zarządzać połączeniem z siecią komputerową. Przesłanie komunikatu przez sieć wymaga bowiem także udziału odbiorcy. Potrzebny jest więc system operacyjny działający na wszystkich połączonych ze sobą komputerach i dbający o prawidłowy przepływ danych między nimi. Taki system nazywamy sieciowym systemem operacyjnym.

Praca z sieciowym systemem operacyjnym przypomina pracę sekretarki, która chcąc załatwić pewną sprawę wysyła listy do odpowiednich instytucji i przeprowadza rozmowy z odpowiednimi osobami. Sekretarka musi przy tym znać wszystkie potrzebne adresy i numery telefonów oraz wiedzieć, gdzie załatwia się konkretną sprawę. Podobnie użytkownik sieciowego systemu komputerowego musi wiedzieć, gdzie znajdują się interesujące go dane i do których komputerów są przyłączone interesujące go urządzenia. Jeśli komputer, na którym znajdują się potrzebne dane, popsuje się, użytkownik nie może zrealizować swojego zadania, podobnie jak sekretarka staje się bezradna, gdy nie nadchodzi list z ważną informacją. Cóż, tak czasem bywa, gdy musimy polegać na innych.

Spróbujmy inaczej

Z tym poleganiem bywa jednak różnie. Jeśli zamierzamy powierzyć jakąś sprawę określonej firmie i mamy do wyboru małą firmę jednoosobową, a drugą dużą zatrudniającą wiele osób, to rzecz jasna bardziej będziemy polegać na firmie dużej. Dlaczego? A choćby dlatego, że czasowa absencja pracownika dużej firmy nie powinna mieć większego wpływu na załatwienie naszej sprawy, czego nie można powiedzieć o firmie jednoosobowej.

Sieciowy system operacyjny pozwala każdemu użytkownikowi komputera w sieci na świadome wykorzystywanie zasobów innych komputerów. Jednak na sieć komputerową można spojrzeć również w zupełnie inny sposób. Oto mamy nie jeden komputer, lecz cały zespół komputerów z możliwością wzajemnej komunikacji. Gdybyśmy mogli nasze zadania zlecać do wykonania nie jakiemuś określonemu komputerowi, ale sieci komputerów jako całości, to uzyskalibyśmy system mający podobną przewagę nad pojedynczym komputerem, jaką ma firma wieloosobowa nad jednoosobową.

Pomysł wydaje się prosty, a korzyści ogromne. Współpracujące komputery mogłyby przekazywać sobie zadania w celu wyrównania wzajemnych obciążeń. W przypadku awarii jednego komputera wykonywane na nim zadania mogłyby być przenoszone na inne komputery. Dane mogłyby być powielane na różnych komputerach, aby w przypadku awarii jednego z nich zawsze była gdzieś dostępna inna kopia. Zadania złożone z wielu współbieżnie wykonujących się procesów mogłyby być wykonywane jednocześnie na wielu komputerach, przez co skróciłby się czas ich realizacji. Ogólnie mówiąc uzyskalibyśmy w ten sposób zrównoleglenie przetwarzania, znacznie większą niezawodność systemu i bardziej równomierne wykorzystanie zasobów.

System operacyjny zarządzający takim zespołem komputerów musi pełnić znacznie więcej funkcji niż zwykły sieciowy system operacyjny. Nie wystarczy bowiem samo nadzorowanie przepływu informacji. Zadania takiego systemu przypominają obowiązki menedżera zarządzającego wieloosobową firmą, który musi rozdzielić prace w sposób właściwy, ustalić zastępstwa na wypadek choroby, określić jakie dokumenty, w ilu kopiach i gdzie będą przychowywane itd.

Choć nie wymyślono jeszcze programów zastępujących menedżerów, istnieją już różne próby opracowania systemów operacyjnych, które zarządzałyby wieloma komputerami połączonymi w sieć, i to w taki sposób, aby stworzyć użytkownikowi

wrażenie, że korzysta on z jednego dużego systemu. Tego typu systemy operacyjne nazywa się rozproszonymi systemami operacyjnymi. W Polsce nie ma jeszcze żadnego systemu tego typu. Na świecie tworzy się je głównie w ośrodkach akademickich, gdzie dziś mają jeszcze raczej charakter eksperymentalny.

Trudności

Istnieje kilka przyczyn, z powodu których rozproszone systemy operacyjne tak wolno zdobywają rynek. Podstawową jest fakt, że nie rozumiemy jeszcze do końca, jak naprawdę działają systemy rozproszone. Są one bardzo złożone – obejmują komputery o różnych architekturach, charakteryzują się dużym stopniem współbieżności, są w nich rozwiązywane bardzo skomplikowane problemy decyzyjne. Trudno przewidzieć wszystkie efekty połączenia ze sobą różnych komputerów (wzajemny wpływ poszczególnych składników, sposób rozprzestrzeniania się błędów, efekt przyłączenia nowych komputerów itd.). Złożoność problemu nie wzrasta tu liniowo wraz ze wzrostem liczby komputerów, podobnie jak problem ułożenia rozkładu jazdy pociągów w dużym węźle komunikacyjnym nie sprowadza się do problemów ułożenia rozkładu jazdy dla każdego pociągu z osobna. Trzeba tu dodatkowo uwzględnić różne wzajemne powiązania. Choć istnieją dowody poprawności algorytmów rozproszonych, są one jednak przeprowadzane przy określonych teoretycznych założeniach i jest trudno stwierdzić, czy rzeczywisty system je spełnia. Te wszystkie czynniki odstraszają potencjalnych użytkowników systemów rozproszonych, którzy nie chcą iść na zbyt duże ryzyko. Jeden ze znanych badaczy algorytmów rozproszonych, L. Lamport, podał taką oto definicję: *System rozproszony to taki, który nie daje ci wyników wtedy, gdy padnie komputer, o którego istnieniu nigdy nie słyszałeś.*

To oczywiście żart. Prace nad systemami rozproszonymi trwają nadal, coraz lepiej je rozumiemy i coraz więcej korzyści będziemy mieli z ich stosowania. Istnieje szansa, że już w niedalekiej przyszłości pojawią się pierwsze wersje komercyjne tego typu systemów.

Co to znaczy rozproszony?

Co to jest system rozproszony? Scentralizowany system liczący to ogólnie mówiąc sprzęt oraz oprogramowanie, czyli sterowanie oraz dane. Rozproszony system liczący to rozproszony sprzęt (a więc co najmniej dwa komputery, każdy z własną pamięcią i procesorem, połączone siecią), rozproszone sterowanie (decyzje są podejmowane hierarchicznie lub autonomicznie) oraz rozproszone dane (rozmieszczone na różnych komputerach, być może w wielu kopiach). Zadaniem rozproszonego systemu operacyjnego jest zarządzanie takim systemem w ten sposób, aby dla użytkownika wyglądał on jak zwykły system scentralizowany. Z każdego komputera powinien więc być do niego taki sam dostęp. Użytkownik nie powinien wiedzeć, gdzie faktycznie znajdują się oraz w ilu egzemplarzach są dane, z których korzysta. Nie musi być on świadomy, gdzie rzeczywiście wykonuje się jego program. Nie powinno go również obchodzić, że w systemie wykonują się jakieś inne zadania oraz niektóre łącza lub komputery mogą się popsuć.

Cechą charakterystyczną dobrego systemu rozproszonego jest brak „pięty Achillesowej", a więc takiego elementu, którego zepsucie się powodowałoby niemożność pracy całego systemu. Dotyczy to w szczególności sprzętu i takich wydarzeń losowych, jak awaria zasilania, pożar lub trzęsienie ziemi. To wymaganie może wydawać się zbyt wygórowane, ale niektórym instytucjom, takim jak wojsko czy banki, bardzo zależy na tego typu niezawodności. Niezawodny system rozproszony powinien więc być rozproszony geograficznie.

Ponieważ od rozproszonych systemów operacyjnych wymaga się znacznie więcej niż od systemów scentralizowanych, trzeba w nich używać specyficznych metod i technik. W dalszej części artykułu omówimy krótko kilka takich technik oraz podamy problemy, jakie się z nimi wiążą.

Powielanie danych

Jednym z typowych mechanizmów stosowanych w systemach rozproszonych jest powielanie danych. Zwiększa ono wydajność systemu i skraca czas jego reakcji, gdyż dostęp do bliżej położonej kopii potrzebnych danych jest szybszy i mniej obciąża sieć komunikacyjną. Powielanie takich danych, które tylko się odczytuje, nie przysparza problemów. Zawsze można stworzyć w dowolnym miejscu nową kopię, tak samo jak zawsze można wywiesić jeszcze jeden egzemplarz ogłoszenia bez względu na to, gdzie i ile ogłoszeń już wywieszono. Jeśli jednak powielona informacja moża się zmieniać, powstaje niebanalny problem zapewnienia jej zgodności z pozostałymi kopiami. Moglibyśmy wymagać, aby ten, kto zmienia dane w swojej kopii, robił to także we wszystkich innych kopiach, uprzedzając o tym wcześniej ich użytkowników. Rozwiązanie takie jest jednak gorsze niż korzystanie z jednej wspólnej kopii. Trzeba więc znaleźć jakiś kompromis między dostępnością a zgodnością, akceptując w niektórych sytuacjach kontrolowaną niezgodność kopii. Zauważmy, że książka telefoniczna jest nieaktualna już w chwili jej wydania. Zgadzamy się jednak na to, bowiem w przeciwnym wypadku nigdy nie można by jej wydać.

Jedną z metod stosowanych w systemach rozproszonych jest pamiętanie dla każdej kopii czasu jej ostatniej modyfikacji. Do odczytania informacji wystarczy mieć dostęp do aktualnej kopii. Niestety na ogół nie wiadomo, która z nich jest aktualna. Trzeba mieć zatem dostęp do takiej liczby kopii, aby jedna z nich na pewno była aktualna. Okazuje się, że przy odpowiednim zarządzaniu nie muszą to być wszystkie kopie, a tylko pewien ich podzbiór. Możemy na przykład wymagać, aby ten kto chce modyfikować dane, miał dostęp do większości kopii. Gwarantuje to, że nigdy dwóch użytkowników nie będzie modyfikowało danych jednocześnie i że wśród modyfikowanych kopii zawsze będzie co najmniej jedna kopia aktualna.

Uniezależnianie się od innych

W systemach scentralizowanych elementem ograniczającym wydajność są przede wszystkim urządzenia pamięci dyskowej, zwykle o kilka rzędów wolniejsze od procesora i pamięci operacyjnej. Sieć komunikacyjna w systemach rozproszonych jest urządzeniem jeszcze wolniejszym od dysku, a w dodatku znacznie bardziej zawodnym. W celu uniezależnienia się do pewnego stopnia od informacji pochodzącej z sieci, a zarazem w celu przyspieszenia pracy systemów rozproszonych, używa się specjalnych mechanizmów, takich jak wskazówki (ang. *hints*), pamięć buforowa (ang. *cache*) i zabezpieczanie (ang. *stashing*).

Wskazówki są to zapamiętane wyniki niektórych operacji wymagających dostarczanych z zewnątrz argumentów. Zapamiętuje się je po to, aby tych operacji nie trzeba było za każdym razem wykonywać, co wiązałoby się z oczekiwaniem na argumenty z sieci. Wskazówka ma jednak taką cechę, że jeśli jest nieaktualna (bo zmieniły się argumenty operacji), to fakt ten zostaje od razu stwierdzony i operację wykonuje się od nowa. Książka telefoniczna jest przykładem zbioru takich wskazówek. Jeśli numer jest nieaktualny, szybko się o tym dowiadujemy, a korzystając z biura numerów możemy dotrzeć do właściwego odbiorcy.

Jest też inne rozwiązanie. Można w ogóle nie kupować książki telefonicznej, a używane numery zapisywać sobie w notatniku.

Za każdym razem, gdy jest potrzebny nowy numer, można zadzwonić do biura numerów. Jeśli po jakimś czasie w podręcznym notatniku zabraknie miejsca, wykreśla się z niego dawno nie używane numery. Notatnik jest tu odpowiednikiem pamięci buforowej, którą w systemach rozproszonych stosuje się w celu przyspieszenia operacji wymagających odwołania do innych komputerów. Z pamięcią buforową wiąże się problem zgodności jej zawartości z oryginałem. Rozwiązuje się go w ten sposób, że o każdej zmianie w oryginale informuje się komputery, które mają w swej pamięci buforowej właśnie zmienianą informację z oryginału. Inaczej mówiąc, każdy kto zmienia numer telefonu, musi powiadomić o tym wszystkich, którzy do niego poprzednio dzwonili.

Pamięć buforową stosuje się także w systemach scentralizowanych, ale służy ona tam do przyspieszenia operacji dyskowych i w związku z tym realizuje się ją w pamięci operacyjnej. W systemach rozproszonych pamięć buforową, przechowującą informację uzyskaną z sieci, można zrealizować nawet na zwykłym dysku, który i tak jest od sieci znacznie szybszy.

Jeszcze inną techniką stosowaną w systemach rozproszonych jest zabezpieczanie. Przypomina ono nasze postępowanie np. przy wysyłaniu listu do osoby, której aktualnego adresu nie jesteśmy pewni. Jeśli nie mamy możliwości ustalenia jaki jest obecny adres, wysyłamy list pod znanym nam dotychczas adresem, mając nadzieję, że w przypadku pomyłki list albo wróci, albo zostanie przekazany dalej. Zabezpieczanie polega na tym, że każdym razem, gdy jest potrzebna informacja od innego komputera, wysyła się do niego żądanie, a odpowiedź zapamiętuje na wszelki wypadek. Jeśli na kolejne żądanie nie otrzyma się odpowiedzi, wykorzystuje się poprzednio zapamiętaną odpowiedź, mając nadzieję, że od tego czasu nie było żadnych zmian.

Granice cierpliwości

Stwierdzenie, że nie ma odpowiedzi na jakieś zapytanie, wymaga również specjalnego postępowania. Komputery komunikują się ze sobą wysyłając komunikaty przez sieć. Komunikaty te mogą zaginąć albo na skutek błędów w łączach, albo w wyniku awarii komputerów pośredniczących w przekazywaniu komunikatu. Nie wiadomo przy tym, czy zaginęło zapytanie, czy odpowiedź. Sytuacja jest dokładnie taka sama, jak w przypadku komunikowania się za pomocą listów. Kiedy bowiem uznajemy, że nie otrzymaliśmy odpowiedzi na nasz list? Zależy to oczywiście od naszej tolerancji w stosunku do poczty oraz od tego, z jakiej odległości ma nadejść odpowiedź, ale zawsze istnieje określona granica cierpliwości. W systemach rozproszonych również ustala się tego rodzaju próg, zwany po angielsku *timeout*, a określający maksymalny czas oczekiwania na odpowiedź. Po upływie tego czasu ponawia się komunikat.

Podobnie jak bywa to z pocztą, komunikaty mogą opóźniać się, ale także mogą przychodzić w innej kolejności niż zostały nadane. Odbiorca musi więc odróżniać nowe żądanie od powtórzenia żądania starego, powinien także odtworzyć prawidłową kolejność komunikatów. W tym celu, podobnie jak na poczcie, stosuje się metodę stemplowania komunikatów czasem (ang. *timestamps*). Ale w przypadku systemów rozproszonych sprawa nie jest tak prosta, jakby się wydawało.

Kłopoty z czasem

Załóżmy, że otrzymaliśmy dwa listy – jeden z Paryża, nadany o godz. 10.00, a drugi z Londynu, nadany o godz. 9.30. Który z nich uznany za wcześniejszy? Na podstawie stempla pocztowego musimy uznać za wcześniejszy list z Londynu. Skądinąd jednak wiemy, że wskutek różnicy czasu, gdy w Londynie była 9.30, to w Paryżu – już 10.30. Odnosimy się tu do pojęcia umownego globalnego czasu, według którego są ustawione wszystkie zegary świata.

W systemie rozproszonym brak jest takiego globalnego czasu. Każdy komputer ma własny zegar, zegary te jednak łatwo rozsynchronizowują się i nie wiadomo, który z nich wskazuje prawidłowy czas. Ponowne zsynchronizowanie zegarów wymaga komunikowania się komputerów między sobą, ale na niewiele to się zdaje, bo przecież nie wiadomo, jak długo komunikat płynie z jednego komputera do drugiego. Zamiast zegarów fizycznych stosuje się wic zegary logiczne. Każdy komputer ma swój taki zegar (może to być zmienna całkowita), który zwiększa po każdej wykonanej operacji. Gdy od innego komputera otrzyma komunikat stemplowany czasem późniejszym niż pokazuje jego lokalny zegar, oznacza to, że coś jest nie w porządku, bo przecież komunikat nie może przyjść wcześniej niż został nadany. Lokalny logiczny zegar trzeba wówczas przestawić na czas co najmniej o jedną jednostkę późniejszy niż w odebranym komunikacie. Takie „doszlusowywanie" do szybszych zegarów pozwala na ich podstawie ustalić właściwą kolejność zdarzeń w całym systemie rozproszonym.

Tworzenie nazw

Niebanalnym problemem jest ustalanie nazw obiektów w systemie rozproszonym oraz określanie ich lokalizacji. Należy pamiętać, że ze względu na efektywność działania systemu obiekt może być przemieszczany z jednego komputera na inny, może być także powielany. Rozwiązanie takie, w którym wszystkie komputery znają lokalizację wszystkich obiektów, jest niepraktyczne, gdyż każda zmiana wymagałaby udziału wszystkich komputerów. Zarządzaniem nazwami zajmują się więc specjalne procesy wykonywane na różnych komputerach i komunikujące się ze sobą w celu ustalenia faktycznego i lokalizacji obiektu. Wygenerowanie dwóch różnych nazw na jednym komputerze nie jest trudne – wystarczy, aby częścią tej nazwy był bieżący czas. W systemie rozproszonym nie ma wspólnego zegara, może się więc zdarzyć, że na dwóch różnych komputerach zostaną wygenerowane te same nazwy. System musi je jednak jakoś rozróżnić. Ponieważ każdy komputer ma swoją nazwę i nazwy te muszą być różne, wystarczy do nazwy obiektu dopisywać nazwę komputera, na którym ten obiekt powstał. Podejście takie ma jednak określone wady. Po pierwsze, wiąże ono obiekty z komputerami, a jak wiemy, obiekty mogą być powielane lub przenoszone z komputera na komputer. Po drugie, komputery o różnych architekturach mogą generować nazwy różnej postaci i długości (zależnej od długości słowa komputera). W systemie rozproszonym trzeba zatem ustalić określony standard globalnych nazw i zasadę przydzielania ich lokalnym obiektom. Takie podejście pozwala dowolnie wybierać nazwy lokalne, a jednocześnie odwzorowywać wiele nazw lokalnych na jedną globalną (lokalne kopie tego samego pliku powinny globalnie nazywać się tak samo).

Gdy trzeba działać wspólnie

Wykonanie operacji, w której musi uczestniczyć jednocześnie kilka komputerów, także nie jest proste. Nie wystarczy każdemu komputerowi wysłać rozkaz „zrób swoje". Nie wiemy bowiem, czy każdy z tych komputerów rzeczywiście będzie mógł „zrobić swoje". Może przecież być właśnie w naprawie lub jego lokalny stan może być taki, że wykonanie operacji zakończy się błędem (np. dzielenie przez zero). Jeśli operacja ma być albo wykonana w całości, albo w ogóle nie być wykonana, to w przypadku błędu na jednym komputerze trzeba odwołać wykonanie operacji na pozostałych. Co jednak zrobić, jeśli ktoś już skorzystał z wyniku tej operacji? A jeśli na podstawie tego wyniku stworzono dane, z których skorzystał ktoś następny? Jak z tego widać, problem

nie jest prosty. Rozwiązuje się go za pomocą specjalnego dwufazowego protokołu, który w pierwszej fazie wymaga zgody wszystkich komputerów na wykonanie operacji, a dopiero w drugiej poleca tę operację wykonać.

Migracja

Bardzo wiele problemów wiąże się z mechanizmem migracji procesów. Jest to jeden z kluczowych mechanizmów systemów rozproszonych, który nie występuje w systemach scentralizowanych. Migracja, czyli przenoszenie procesów z jednego komputera na drugi, jest narzędziem służącym do równoważenia obciążenia systemu komputerowego, a także pozwalającym zwiększyć równoległość wykonywania programów aplikacyjnych. Narzędzie to umożliwia również wykonywanie procesów na tych komputerach, które mogą najlepiej zaspokoić wymagania tych procesów, oraz zwiększa niezawodność systemu, gdyż w przypadku awarii jednego komputera pozwala „uciec" z obliczeniami na inny komputer. Zrealizowanie tego mechanizmu nie jest jednak proste. Przede wszystkim trzeba umieć odpowiedzieć na następujące podstawowe pytania: kiedy, dokąd i który proces przenosić. W tym celu należy zbierać dane o zachowaniu się systemu. Dla każdego komputera może to być np. liczba wykonywanych procesów, obciążenie procesora, liczba wysłanych komunikatów itp. Trzeba również potrafić określić stan procesu, czyli zgromadzić o nim, wszystkie te informacje, które mają być przenoszone.

Istotne jest również to, kiedy przenoszony proces może zacząć wykonywać się. Czy dopiero po przeniesieniu całości, czy też może wcześniej, gdy zostanie przeniesiona tylko informacja niezbędna do dalszego wykonywania. W systemie heterogenicznym, tzn. złożonym z komputerów o różnej architekturze, należy uwzględnić różną reprezentację danych. Nie ma na przykład sensu przenoszenie danych, jeśli cecha liczb zmiennopozycyjnych na maszynie docelowej ma mniej bitów, niż na maszynie wyjściowej, ponieważ obcięcie bitów cechy zmienia całkowicie wartość liczby.

Trzeba także zdecydować, co robić z komunikatami, które będą kierowane do przeniesionego już procesu. Jest interesujące, że stosowane w tym przypadku rozwiązania odpowiadają dokładnie temu, co ludzie zwykli robić przeprowadzając się do innej miejscowości czy kraju. Są trzy podejścia. W pierwszym, proces pozostawia w miejscu poprzedniego pobytu informacje o tym, jaki jest jego nowy adres. Komunikaty docierające pod stary adres są kierowane dalej pod nowy adres. To rozwiązanie jest dość kłopotliwe, zwłaszcza wtedy, gdy proces często przenosi się. Drugie podejście wymaga, aby migrujący proces jeszcze przed przeniesieniem wysłał do wszystkich komunikujących się z nim procesów informację o nowym adresie. Będą one więc od razu wysyłać komunikaty tam, gdzie trzeba. Trzecie podejście stosuje się w przypadku nagłych przenosin (np. wtedy, gdy pada komputer, na którym proces wykonywał się). Informacja o nowym adresie jest rozsyłana dopiero z nowego miejsca. W ten sposób część komunikatów wysłanych pod stary adres może przepaść. Komunikujące się procesy muszą to rozpoznać i ewentualnie powtórzyć niektóre komunikaty, wysyłając je tym razem pod nowy adres.

Uwaga na intruzów

Istotnym problemem jest także ochrona systemów rozproszonych przed atakami osób nieuprawnionych. W systemach scentralizowanych główny nacisk jest położony na ochronę plików, chodzi bowiem o to, aby jeden użytkownik nie mógł dostać się do danych czy programów innego użytkownika, jeśli nie został do tego uprawniony. W systemach rozproszonych należy dodatkowo zapewnić bezpieczną komunikację między komputerami. Zadanie to nie jest łatwe, gdyż potencjalny napastnik lub intruz ma tu wiele możliwości. Na atak jest narażony praktycznie każdy element systemu znajdujący się na drodze komunikatu. Dotyczy to zarówno sprzętu, jak i oprogramowania. Komunikaty można bowiem przechwytywać programowo ingerując w tę część systemu operacyjnego, która zajmuje się przesyłaniem komunikatów, ale można także przyłączyć się do fizycznych łączy i podsłuchiwać je tak, jak podsłuchuje się rozmowę telefoniczną. Co więcej, okazuje się, że wiele informacji można uzyskać z samego tylko badania zmian pola elektromagnetycznego wokół sprzętu komputerowego.

Ataki i obrona

Rozróżnia się dwa podstawowe typy ataków na system komunikacyjny. Ataki pasywne, które polegają wyłącznie na przechwyceniu komunikatu lub rozpoznaniu niektórych własności statystycznych strumienia komunikatów takich, jak np. ich częstotliwość oraz długość. Jeśli np. napastnik zna położenie komunikujących się procesów i wie, że toczy się jakaś negocjacja, to informacje o długości i częstotliwości komunikatów mogą być bardzo istotne. Obroną przed atakami pasywnymi jest przede wszystkim szyfrowanie komunikatów i to w taki sposób, aby ukryć własności statystyczne szyfrowanej informacji.

O ataku aktywnym mówimy wtedy, gdy napastnik wykonuje jakąś operację na komunikacie. Może to być zmiana zawartości komunikatu, zmiana kolejności komunikatów, wstawienie dodatkowych komunikatów, odmowa dalszego przesłania komunikatu lub jego przesłanie w inne miejsce niż należało, opóźnienie dostarczenia komunikatu. Napastnik może także podsłuchać wymianę komunikatów inicjującą nawiązanie rozmowy między dwoma procesami, a później nadać te same komunikaty, podając się za kogoś innego. Dobry system ochrony powinien umożliwiać szybkie rozpoznanie ataku aktywnego oraz naprawienie wynikających z niego szkód. W tym celu stosuje się skomplikowane protokoły komunikacyjne, które oprócz szyfrowania wymagają także kontroli błędów parzystości, numerowania komunikatów, stosowania specjalnych podpisów cyfrowych oraz dodatkowej identyfikacji nadawcy i odbiorcy.

Szyfrowanie

Szyfrowanie jest także istotnym problemem, gdyż stosowany szyfr powinien być praktycznie nie do złamania, nawet przy użyciu najszybszych metod na najszybszych komputerach. Szyfry takie opierają się na znajomości tajnego klucza szyfrowania. Sam klucz musi być jednak przekazany drugiej stronie i nie może być zakodowany, a więc musi być przekazany bezpiecznym łączem. Problemu tego można uniknąć stosując system kodowania oparty na tzw. kluczach publicznych. W systemie tym do zaszyfrowania informacji służą powszechnie dostępne klucze publiczne, a do jej odszyfrowywania tajne klucze prywatne. Klucze te mają tę własność, że na podstawie klucza prywatnego można łatwo wyznaczyć klucz publiczny, ale operacja odwrotna jest praktycznie niewykonalna. Wydaje się to dziwne, bo dla każdej funkcji można znaleźć funkcję do niej odwrotną. Tak jest i w tym przypadku. Aby znaleźć funkcję odwrotną trzeba rozłożyć na czynniki pierwsze pewną dużą liczbę naturalną. Ogólnie wiadomo, jak to się robi. Okazuje się jednak, że dla rzeczywiście dużych liczb najlepszy algorytm (a jest to algorytm niewiele różniący się od algorytmu znanego już ze szkoły podstawowej) na najszybszych komputerach będzie to robił niewyobrażalnie długo. System kluczy publicznych nie wymaga przesyłania tajnego klucza – każdy zostawia go dla siebie, a wysyła tylko klucz publiczny, który może być znany wszystkim. Jednak i ten sposób nie jest doskonały. W takim systemie intruz może łatwo podrzucić do szyfrowania swój klucz twierdząc, że należy on do kogoś innego, i w ten sposób przechwycić zaszyfrowane komunikaty.

Powiedz mi kim jesteś

W systemie rozproszonym bardzo istotny jest więc również mechanizm identyfikacji, pozwalający stwierdzić, czy ten kto wysyła komunikat jest rzeczywiście tym za kogo się podaje. Stwierdzenie tego faktu nie jest jednak proste, jeśli zważymy, że identyfikacja także odbywa się za pomocą komunikatów, które mogą być przechwycone. Opracowano wiele metod identyfikacji. Jedna z nich bardzo przypomina nasze postępowanie w przypadku, gdy np. stajemy przed problemem, czy nieznany nam wcześniej młody człowiek jest rzeczywiście synem brata drugiej żony naszego stryjka z Ameryki. Aby to stwierdzić, wystarczy spytać się tego młodzieńca, np. w którym roku umarł wujek Antoś. Podobnie w systemie rozproszonym, chcąc potwierdzić tożsamość nadawcy komunikatu, wystarczy wysłać żądanie podania wyniku operacji, którą tylko oryginalny nadawca może wykonać prawidłowo (operacja taka może np. wymagać użycia tajnego klucza). W tym przypadku nie trzeba więc przesyłać tajnego klucza i wystarczy wykazać, że ma się taki klucz. Nie trzeba też specjalnie kodować komunikatów, jeśli za każdym razem będziemy wymagali wykonania trochę innej operacji.

Systemy rozproszone to przyszłość

Ten krótki przegląd niektórych tylko zagadnień związanych z systemami rozproszonymi pozwala uświadomić sobie, jak bardzo są one złożone i jak bardzo różnią się od zwykłych systemów scentralizowanych.

To wszystko może brzmieć bardzo egzotycznie, zwłaszcza dla tych, którzy uważają za wielki sukces możliwość przyłączenia swego komputera osobistego do sieci typu Novell, a za jeszcze większy – możliwość przyłączenia się do sieci Internet i wymiany informacji z całym światem. Oczywiście sieciowe systemy operacyjne stanowią istotny krok w rozwoju informatyki. Na tym jednak rozwój ten się nie kończy. Jeżeli chcemy coraz bardziej upodabniać komputery do siebie, musimy iść w kierunku systemów rozproszonych. Przecież sami działamy w sposób rozproszony. Stworzyliśmy system społeczny, w którym każdy nie musi umieć wszystkiego, a zarazem w którym nie ma ludzi niezastąpionych. Nasz własny organizm funkcjonuje bardzo sprawnie, mimo że co chwilę ginie tysiące jego komórek, a czasem umiera cały organ.

Wiele naszych problemów wymaga zdecentralizowanego podejmowania decyzji, opartego na współpracy autonomicznych zespołów. Komputerowe systemy rozproszone znacznie lepiej nadają się do rozwiązywania takich problemów niż systemy scentralizowane, gdyż już sama ich struktura lepiej przystaje do struktury problemu. Systemy rozproszone są (w założeniu, choć może jeszcze nie zawsze w praktyce) znacznie bardziej niezawodne niż systemy scentralizowane. Nie chodzi tu tylko o mechanizmy zwiększające niezawodność, takie jak migracja procesów czy powielanie danych. Także utrzymanie pełnej sprawności systemu rozproszonego jest znacznie tańsze niż utrzymanie pełnej sprawności systemu scentralizowanego.

W przypadku tego drugiego trzeba mieć w zapasie jeszcze jeden taki sam system, który można natychmiast włączyć w razie nagłej awarii (taki sposób zabezpieczenia stosują duże banki, również na statkach kosmicznych umieszcza się zapasowe komputery). System scentralizowany musi więc być powielony w 100%. Do utrzymania pełnej sprawności systemu rozproszonego wystarczy jednak ok. 20% dodatkowego sprzętu (a więc np. kilka zapasowych komputerów, które można natychmiast włączyć w sieć, aby w przypadku awarii zachować pełną moc obliczeniową systemu). Odporność na błędy i możliwość równoległego wykonywania obliczeń, to dwie własności systemów rozproszonych powodujące, że mogą one być znacznie silniejsze niż tradycyjne systemy scentralizowane.

Tak jak lata sześćdziesiąte były erą przetwarzania wsadowego, lata siedemdziesiąte erą dużych systemów z podziałem czasu, lata osiemdziesiąte zaś erą komputerów osobistych, tak lata dziewięćdziesiąte będą niewątpliwie erą systemów rozproszonych.

Nowy kwartalnik informatyczny „MACHINE GRAPHICS & VISION"

„Machine GRAPHICS & VISION" oznaczony symbolem ISSN 1230–0535 jest angielskojęzycznym kwartalnikiem naukowym, redagowanym przez międzynarodowe Kolegium Redakcyjne. Czasopismo zamieszcza materiały dotyczące metod syntezy i analizy modeli obliczeniowych, które mogą być podstawą komputerowego generowania, analizowania i przetwarzania obrazów.

Kwartalnik zawiera publikacje materiałów z Konferencji nt. Grafiki Komputerowej i Przetwarzania Obrazów (GKPO) odbywającej się w Polsce każdego parzystego roku (ok. 20 maja), nadsyłane do Redakcji wyniki bieżących prac badawczych, a także definiuje kierunki zainteresowań w prezentowanej dziedzinie. Jest miejscem wymiany myśli, poglądów i wyników badań dotyczących komunikowania się komputera z otoczeniem za pomocą obrazów oraz prezentacji zastosowań tego typu systemów.

Preferowane są opracowania dotyczące syntezy obrazów, prezentowania danych i wiedzy graficznej, wizualizacji sterowanej wiedzą, sztuki wizualnej, geometrii obliczeniowej, trójwymiarowego obrazowania fizycznych, bryłowych i obrazowych modeli oraz ich transformacji, animacji komputerowej, wzmacniania, odnawiania i kompresji obrazów, detekcji brzegów i segmentacji, matematycznych metod analizy obrazów, pasowania wzorców i modeli oraz rozpoznawania obiektów, opisywania, modelowania i rozumienia scen trójwymiarowych, widzenia mono i stereoskopowego, modelowania ludzkiej percepcji wizyjnej, myślenia obrazowego.

Kwartalnik jest dostępny w systemie prenumeraty.
Cena prenumeraty rocznej:
16 USD (lub złotówkowa równowartość) – prenumerata osobowa;
33 USD – prenumerata instytucjonalna (biblioteczna).
Wydawca: Instytut Podstaw Informatyki PAN.
01-237 Warszawa, ul. Ordona 21. tel. (22) 36-28-41, 36-28-62, 36-28-85, 36-37-09 nr 49 lub 35
Redaktor Naczelny:
Wojciech Mokrzycki – adres i telefony j.w.
poczta elektroniczna: wmokrzyc ⓐ plearn. bitnet
faks: (22) 37-65-64.

STANISŁAW WRYCZA
TOMASZ PLATA-PRZECHLEWSKI
Katedra Organizacji Przetwarzania Danych
Uniwersytet Gdański

Badania ankietowe

Ocena poziomu zastosowań informatyki dla celów zarządzania w Polsce

W ubiegłym roku Katedra Organizacji Przetwarzania Danych Uniwersytetu Gdańskiego w ramach grantu KBN 1 1416 91 02 przeprowadziła badania na temat *Ocena poziomu zastosowań informatyki dla celów zarządzania w Polsce. Badania ankietowe*. Celem tych badań była obiektywna, statystyczna ocena poziomu zastosowań informatyki dla celów zarządzania organizacjami gospodarczymi i instytucjami w Polsce. W zakresie ocen tego poziomu istnieje wiele intuicyjnych opinii wyrażanych przez informatyków. Tylko tego typu reprezentatywne badania mogą nadać im charakter obiektywny. Badania takie podjęto po raz pierwszy w Polsce. Najdłuższa tradycja ich realizacji istnieje w Stanach Zjednoczonych (od 1980 roku) [1]. W ostatnich latach podjęto je również w innych krajach np. w Niemczech, na Tajwanie oraz w Słowenii [3].

Badaniu poddano próbę 500 respondentów – zakładów pracy z całego kraju, zarówno z dużych aglomeracji, jak i z mniejszych miast. Przy wyborze próby zdecydowano się na formę doboru celowego. Kryterium doboru było zainteresowanie współczesnymi kierunkami informatyki. Uwzględniano więc te zakłady pracy, których pracownicy uczestniczyli w różnorodnych konferencjach, sympozjach i wystawach krajowych na temat nowoczesnych metod i narzędzi informatyki. W tym aspekcie metoda doboru celowego wydaje się bardziej adekwatna i obiektywna niż bardziej kosztowna metoda doboru losowego. Stosownie do założeń rozesłano ponad 500 ankiet. Jednakże skuteczność badania pocztowego okazała się bardzo niska (niski współczynnik zwrotów), dlatego zatrudniono ankieterów, którzy instruowali respondentów w trakcie wypełniania ankiet. W rezultacie otrzymano 107 wypełnionych ankiet, których analiza jest zasadniczym przedmiotem badań.

Przyjęte założenia oraz wielkość próby gwarantują dobre oszacowanie dla całości populacji (co jest zasadniczym celem ankiety) oraz dla większości pytań, w których następuje podział populacji na kilka grup. Przy wyodrębnieniu większej liczby podpopulacji liczebność próby nie gwarantuje uzyskania dobrych oszacowań [2, 4], co należy uwzględnić przy ich analizie. Analogiczne badania przeprowadzone i zebrane metodą pocztową w USA były oparte na próbie o liczebności od 104 do 126.

Dla przeprowadzenia analizy statystycznej posłużono się pakietem statystycznym SPSS/PC+. Liczba wygenerowanych tablic i zestawień dla wszystkich trzech części pracy jest olbrzymia. Autorzy stanęli przed zagadnieniem doboru właściwego materiału. Kierowano się tu kryterium wyboru najbardziej istotnych trendów i zależności. Pełny materiał statystyczny jest dostępny w Katedrze Organizacji Przetwarzania Danych Uniwersytetu Gdańskiego.

Najbardziej istotne wyniki badań zostaną przedstawione w cyklu artykułów zamieszczonych w kolejnych numerach INFORMATYKI.

Rezultaty badań mają znaczenie poznawcze przede wszystkim dla profesjonalistów kreujących rozwój informatyki w Polsce. Określone znaczenie praktyczne badania tę będą miały również dla agend rządowych oraz funkcjonujących na rynku firm sprzętowych, software'owych oraz doradczych. Badania te stanowią dobry punkt startu do ich cyklicznej realizacji. Na przykład w USA i w Słowenii powtarza się je co trzy lata.

Kwestionariusz ankiety

Jako metodę realizacji badań wybrano ankietę złożoną z dwóch części. Pierwszą, pn. *Informatyka w zakładzie pracy* można nazwać częścią identyfikującą. Pozwalała ona oszacować wielkość zakładu respondenta (organizacja gospodarcza lub instytucja), liczebność służb informatycznych, wyposażenie w sprzęt i oprogramowanie, powiązania informatyki ze strategią respondenta oraz jego ogólną kulturę informatyczną. Druga część ankiety obejmowała 25 głównych zagadnień tworzenia i eksploatacji systemów informatycznych u respondenta. Każde z tych zagadnień zostało ocenione w skali od 1 do 10, stosownie do oceny jego znaczenia dla aktualnego i przyszłego stanu rozwoju informatyki w ankietowanym zakładzie. Wyjaśnienia zamieszczone w drugiej części ankiety mają również walor poznawczy.

Część pierwsza ankiety (Informatyka w zakładzie pracy)

Część ta zawiera następujące pytania:

1. Liczba osób zatrudnionych w zakładzie pracy;
2. Rodzaj (11 rodzajów) prowadzonej przez zakład pracy działalności;
3. Liczba osób zatrudnionych w informatyce;
4. Liczba osób zajmujących się tworzeniem systemów informatycznych (analitycy, projektanci, programiści);
5. Wykształcenie osób tworzących systemy informatyczne (cztery typy wykształcenia);
6. Zwierzchnik działu informatyki (trzy rodzaje szczebli kierownictwa zakładu);
7. Łączna liczba terminali oraz liczba PC w zakładzie pracy (w tym używanych przez osoby tworzące systemy informatyczne);
8. Czy zakład pracy sformułował plan informatyzacji przy użyciu BSP (*Business System Planning*) lub innej równoważnej metody (pięć typów ustosunkowania się do tego problemu)?

9. Czy w zakładzie pracy istnieje formalna procedura pomiaru poziomu satysfakcji użytkowników z usług i produktów działu informatyki (tak, nie)?
10. Czy są oceniane koszty tworzenia systemów informatycznych w zakładzie pracy i czy się nimi steruje (rodzaj używanej metody oceny lub brak oceny)?
11. Czy rejestruje się i analizuje koszty tworzenia pakietów zastosowań na określone potrzeby pracowników zakładu (tak, nie)?
12. Czy rejestruje się i analizuje koszty eksploatacji istniejących systemów (tak, nie)?
13. Które z następujących podejść do tworzenia systemów informatycznych są używane przez twórców systemów informatycznych i jak często? Jaki jest czas (roboczogodziny w %) stosowania każdego z tych podejść?
a) podejście tradycyjne (klasyczne),
b) podejście oparte na cyklu życia systemu,
c) podejście strukturalne,
d) podejście zautomatyzowane,
e) prototypowanie,
f) zakup i adaptacja gotowego oprogramowania,
g) inne podejścia (jakie).
14. Czy zakład pracy używa narzędzia CASE i jakiego?
15. Czy i jak często używasz JAD (*Joint Application Design*)?
16. Czy i jakie są używane narzędzia reengineering?
17. Przybliżona liczba nieinformatyków używających komputera:
a) na poziomie menu (wprowadzanie danych, wyszukiwania danych, generowanie zdefiniowanych uprzednio raportów),
b) na poziomie komend do zapytań i zestawień (potrafią użyć wyrażeń dBase, SQL, SAS,...),
c) samodzielnie tworzą niewielkie programy używając arkuszy kalkulacyjnych, narzędzi baz danych (dBase, FoxPro,...) lub języków czwartej generacji.
18. Czy zakład pracy ma centrum informacyjne – (formalną grupę lub jednostkę organizacyjną obsadzoną głównie przez informatyków, pomagającą użytkownikom samodzielnie tworzącym oprogramowanie z zakresu ich działalności (tak, nie)?
19. Jak w skali dziesięciopunktowej oceniają pracę działu informatyki:
a) pracownicy działu,
b) użytkownicy,
c) dyrekcja zakładu.
20. Jaką politykę zakupu sprzętu i oprogramowania prowadzi zakład pracy:
a) wykorzystując dostępne zasoby kupuje tańszy sprzęt i oprogramowanie, umożliwiające częściową realizację celów firmy,
b) dostępne środki umożliwiają zakup sprzętu i oprogramowania zgodnego z założonymi celami.
21. Zakupując komputery osobiste dyrekcja zakładu kierowała się następującymi kryteriami:
a) rozwiązują wszystkie problemy zakładu,
b) nie zaspokajają potrzeb, ale są łatwo dostępne i istnieje wiele firm świadczących usługi programowe i sprzętowe,
c) nie zaspokajają potrzeb, ale jest dużo taniego i darmowego oprogramowania, częściowo zaspokajającego potrzeby.
22. Jeżeli zakład pracy ma sieć lokalną (jaką), to jaka jest jej ocena:
a) działa bez zarzutu,
b) zdarzają się sporadyczne problemy,
c) awarie są częste,
d) z powodu awarii zaprzestano korzystania z sieci.
23. Ocena w skali dziesięciopunktowej dostępnego na rynku szkolenia w zakresie sprzętu i oprogramowania (liczba oraz jakość szkoleń),
24. Ocena w skali dziesięciopunktowej stopnia zaangażowania dyrekcji zakładu w rozwój systemów informatycznych.
25. Czy dyrekcja zakładu ułatwia rozwój zdolności organizacyjnych pracowników służb informatycznych (ocena w skali dziesięciopunktowej)?
26. Kryteria używane przy pozyskiwaniu nowych systemów informatycznych (tak, nie):
a) cena,
b) jakość,
c) licencja producenta,
d) okres gwarancyjny,
e) opinia o producencie,
f) opinia innych użytkowników.
27. Przyczyny niepowodzeń we współpracy z partnerami zewnętrznymi (łączenie z zagranicznymi) w dziedzinie systemów informatycznych.

Część druga ankiety (Główne zagadnienia tworzenia i eksploatacji systemów informatycznych)

Zawiera określenie oraz ocenę w skali dziesięciopunktowej następujących zadań uznanych za najważniejsze dla służb informatycznych w zakładzie pracy w ciągu najbliższych trzech do pięciu lat.

1. Tworzenie architektury informacyjnej. W celu zidentyfikowania głównych informacji używanych w zakładzie pracy i ich powiązań z działalnością firmy jest potrzebna kompleksowa architektura informacyjna. Niezbędne jest kierowanie tworzeniem oprogramowania oraz ułatwienie integracji i wspólnego wykorzystania zasobów danych.

2. Usprawnienie strategicznego planowania systemów informatycznych. Dopasowanie strategicznego planu systemów informatycznych do strategicznego planu działalności gospodarczej ma coraz większe znaczenie dla sukcesów firmy. Szybko zmieniające się warunki zewnętrzne, wzrastający poziom zaangażowania użytkowników i przyspieszona zmiana technologiczna podkreślają potrzebę usprawnienia umiejętności planowania strategicznego.

3. Efektywne wykorzystanie zasobów danych. Zasoby danych w zakładzie pracy powiększają się, stają się coraz bardziej skomplikowane, wzrasta ich wartość. Pomimo tego pozostają one w dużym stopniu nie rozpoznane, niedostępne i nie wykorzystane. Dział informatyki musi stworzyć u siebie i w całym przedsiębiorstwie klimat, w którym dane będą traktowane jako „kapitał" zakładu pracy.

4. Dobór, rekrutacja i szkolenie kadr informatycznych. Brak wykwalifikowanych kadr (obecnie i w przyszłości) stanowi zagrożenie dla zdolności do bieżącego zaspokajania potrzeb zakładu pracy przez dział informatyki. Należy określić ścieżki awansu. Należy położyć większy nacisk na rozwijanie umiejętności samodzielnego, twórczego działania.

5. Stosowanie systemów informatycznych w celu osiągnięcia przewagi konkurencyjnej. W wielu dziedzinach działalności gospodarczej przetrwanie firmy zależy od zastosowania systemów informatycznych. Stosuje się je w celu osiągnięcia przewagi konkurencyjnej, gdyż bez tego zakład pracy może być wyeliminowany z gry rynkowej. Przewaga konkurencyjna polega tu na prawidłowym rozpoznaniu własnych możliwości i szans przez innowacje oraz szybkie ich wdrażanie.

6. Umiejscowienie działu informatyki w strukturze organizacyjnej zakładu pracy. Efektywność zaspokajania potrzeb informatycznych zakładu pracy zależy od właściwego miejsca służb informatycznych w firmie. Często są one umiejscowione niewłaściwie w strukturze przedsiębiorstwa.

7. Ułatwienie uczenia się i wykorzystania systemów informatycznych przez zakład pracy. Zakłady pracy, które w swojej działalności wykorzystują nowe technologie informatyczne, mają największe szanse na rozwój. W wielu wypadkach będzie to wymagało modyfikacji sposobów działania i struktur organizacyjnych. Kadry informatyczne także muszą wykazać się zdolnością do uczenia się nowych technologii.

8. Rosnące zrozumienie roli i wkładu służb informatycznych. Informatyka jest czasami postrzegana jako nadmierny wydatek. Nie docenia się jej wkładu w działalność gospodarczą zakładu pracy. Może to doprowadzić do tego, że kierownictwo zakładu zacznie stawiać nierealne żądania i jednocześnie będzie obciążać fundusze, przez co firma straci potencjalne szanse i możliwości.

9. Planowanie i wdrażanie systemów telekomunikacyjnych. Łączność jest krwioobiegiem zakładu pracy. Wykorzystanie informatyki dla uzyskania przewagi konkurencyjnej w dużym stopniu zależy od telekomunikacji. Duże i gwałtowne zmiany w przemyśle zadanie to komplikują.

10. Poprawa jakości tworzenia oprogramowania. Zwłoka w tworzeniu oprogramowania utrzymuje się wciąż na niedopuszczalnie wysokim poziomie. Użytkownicy są zniecierpliwieni, a jeśli uwzględnimy również rosnące koszty osobowe, to potrzeba poprawy efektywności tworzenia systemów staje się oczywista.

11. Umożliwienie elektronicznej wymiany danych oraz integracji różnych środowisk sprzętowych i programowych. Komunikacja elektroniczna z dostawcami i klientami może dać firmie dużą przewagę konkurencyjną. Kierownictwo służb informatycznych powinno nalegać na tworzenie i stosowanie standardów. Standardy umożliwią elektroniczną wymianę danych oraz efektywne funkcjonowanie środowisk sprzętowych i programowych pochodzących od różnych dostawców bez większych kosztów konwersji i utraty efektywności.

12. Planowanie i kierowanie rozwojem zestawu systemów informatycznych zakładu pracy. Gwałtownie rosną rozmiary, złożoność i koszty eksploatacji zestawu systemów informatycznych. Pomimo, że problem ten jest znany od dawna, zbyt mało wiadomo o skutecznym jego rozwiązywaniu.

13. Pomiar efektywności i wydajności systemów informatycznych. Pomiar skuteczności systemów informatycznych ma zasadnicze znaczenie dla efektywnego zarządzania zakładem pracy. To zagadnienie staje się coraz bardziej istotne, w miarę wzrostu inwestycji w systemy informatyczne.

14. Integracja przetwarzania danych, automatyzacji prac biurowych, telekomunikacji i automatyzacji procesów produkcyjnych. Istnieje możliwość integrowania systemów opartych na różnorodnych technologiach. W miarę ich integracji pojawiają się różne problemy organizacyjne.

15. Poprawa bezpieczeństwa i kontroli informacji. W miarę wzrostu uzależniania się zakładu pracy od systemów informatycznych rośnie ryzyko zmiany, zniszczenia, czy też ujawnienia informacji na zewnątrz, a nawet zniszczenia systemu informatycznego. Koniecznością staje się ścisła kontrola bezpieczeństwa danych i odporności systemów na błędy przetwarzania.

16. Stwarzanie warunków i zarządzanie przetwarzaniem danych przez użytkownika. Rozpowszechnianie się samodzielnego przetwarzania danych przez użytkownika za pomocą komputerów osobistych daje szansę wzrostu wydajności, ale stwarza także zagrożenia wynikające z niewłaściwego zarządzania. Kierownictwo służb informatycznych powinno wyważyć proporcje między wymaganiami zarządzania, a swobodą użytkowników. Niezbędne jest określenie roli służb informatycznych i użytkowników.

17. Tworzenie i zarządzanie systemami rozproszonymi. Systemy rozproszone stawiają szereg wyzwań dotyczących ustanowienia i utrzymania spójnych danych, łączenia komputerów pracujących w różnych sieciach, reagowania na zwiększone zapotrzebowanie na wspólne z użytkownikiem tworzenie systemów, zarządzanie, zbędne powielanie wysiłków ustanawianiem zasad udostępniania zasobów danych różnym użytkownikom.

18. Planowanie i zastosowanie technologii CASE. Na całym świecie można zaobserwować wyraźny postęp w automatyzacji funkcji gospodarczych. Można jednak bardziej zwiększyć wydajność, jeżeli sam proces tworzenia systemów informatycznych zostanie zautomatyzowany. Wówczas zasadą jest wspieranie przez systemy informatyczne integracji prac projektowych przez standaryzację metodyk i generowanie kodu. Ponieważ technologia ta jest ciągle w stadium rozwoju, wspieranie tworzenia systemów pakietami CASE jest jeszcze zjawiskiem rzadkim i będzie wymagało zasadniczych zmian w funkcjonowaniu działu informatyki.

19. Projektowanie struktury informacyjnej typu pytanie-odpowiedź. Problemy eksploatacji, zmieniającej się struktury przedsiębiorstwa, integracji pakietów, systemów tworzonych przez użytkownika oraz jakości zestawu systemów informatycznych będą wymagały szerokiej automatyzacji funkcjonowania informatyki, niezbędnej do koordynacji działań i zarządzania przedsiębiorstwem. Projektowanie zintegrowanej bazy technologicznej jest kluczem do wydajności firmy.

20. Wykorzystanie systemów informatycznych do wywierania wpływu na strukturę organizacyjną. Powszechnie akceptowany jest fakt, że nowe technologie informatyczne dają możliwość spłaszczenia struktur organizacyjnych, dając w wyniku bardziej elastyczną i innowacyjną organizację. Systemy informatyczne mogą także ułatwić powstanie nowych, dynamicznych form organizacji, poprawiając zdolność firmy do reagowania na nowe sytuacje.

21. Użytkowanie systemów wspomagania kierownictwa. Możliwości wykorzystywania przewagi konkurencyjnej zależą od zdolności kierownictwa zakładu do eksperymentowania z wariantami decyzji. Narzędzia wspomagania decyzji od dawna są postrzegane jako metoda dostarczania kierownictwu narzędzia modelowania w celu pomocy w podejmowaniu decyzji, jednakże, jak dotąd, daje to niejednoznaczne wyniki.

22. Stworzenie efektywnych zdolności wychodzenia z katastrof. Codziennie rośnie ryzyko potencjalnych katastrof. Powinny istnieć efektywne plany zachowania zasobów informatycznych, zabezpieczające regularnie testowane w celu zmniejszenia potencjalnych strat. W miarę wzrostu wielkości i poziomu integracji oprogramowania rośnie również ryzyko ich utraty.

23. Planowanie i stosowanie technologii przetwarzania obrazów. Technologie tą są coraz częściej stosowane przez przedsiębiorstwa. Istnieje potrzeba ich zintegrowania z systemami informatycznymi oraz rozważenia ich wpływu na planowanie technologii, zbieranie danych, konfigurację sprzętu i oprogramowania oraz politykę użytkowników.

24. Przechodzenie w kierunku systemów globalnych. Wraz z rosnącą globalną orientacją rynków, dostawców, klientów

itp., działalność gospodarcza będzie wymagała globalnego planowania, projektowania i wdrażania informacji w celu zapewnienia efektywnych i wydajnych działań. Jest to wyzwanie dla obecnych systemów transferu danych, koncepcji czasu pracy grupowej i stosowania różnych języków.

25. Księgowanie wartości oprogramowania i zasobów danych. Obecnie ponoszone nakłady na tworzenie oprogramowania i zasobów danych nie są amortyzowane. Efektywne długookresowe zarządzanie firmą wymaga odpowiedniej metody księgowania wartości oprogramowania i zasobów danych.

Informatyka w zakładzie pracy

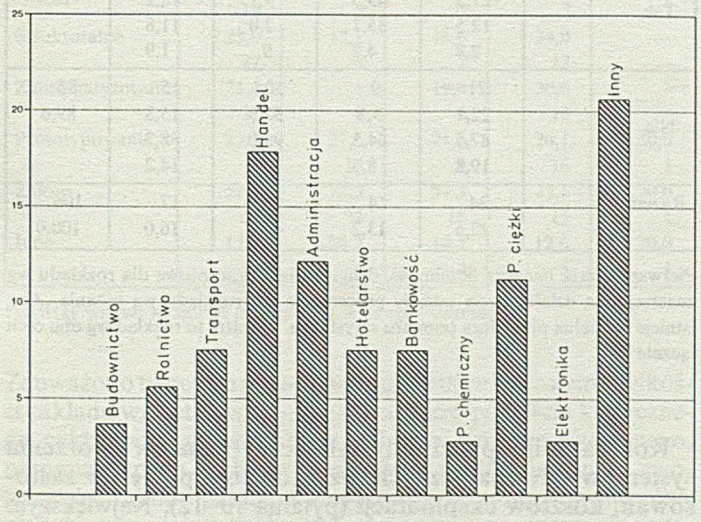
Poniżej zostaną omówione wyniki analizy części pierwszej ankiety. Jak wiadomo, obejmowała ona 27 pytań, pozwalających scharakteryzować próbę pod względem różnych mierników wielkości zakładu pracy oraz stopnia zaawansowania informatyki. Omówienie odbywać się będzie w kolejności występowania w ankiecie pytań.

Pierwszym zagadnieniem była ocena wielkości przedsiębiorstwa, którą charakteryzowano liczbą zatrudnionych ogółem. Rozkład przedsiębiorstw według tego kryterium przedstawia tabela 1.

Tabela 1. Rozkład przedsiębiorstw według liczby zatrudnionych ogółem

Liczba zatrudnionych	Liczba zakładów	Odsetek zakładów	Odsetek kumulowany
0– 10	5	4,7	4,7
10– 50	12	11,2	15,9
50– 100	21	19,6	35,5
100– 500	32	29,9	65,4
500–1000	15	14,0	79,4
1000–5000	17	15,9	95,3
5000 i więcej	5	4,7	100,0
Razem	107	100,0	100,0

W próbie dominują przedsiębiorstwa średnie, natomiast przedsiębiorstwa bardzo małe i wielkie stanowią skrajne wielkości i dlatego ich liczebna reprezentacja w próbie jest niewielka (np: jednostki o zatrudnieniu do 5 osób – niecałe 2%). Średnie zatrudnienie w badanych jednostkach wynosiło 1125 osób (mediana 255 osób, wartość dominująca 70).



Wydruk 1. Rozkład przedsiębiorstw według rodzaju prowadzonej działalności gospodarczej

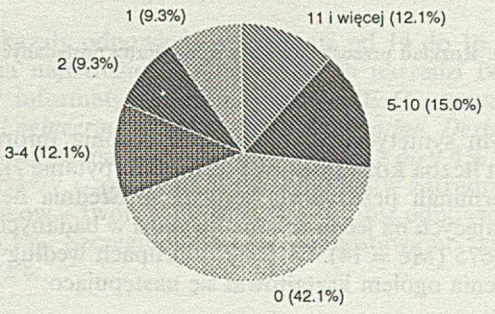
W pytaniu 2. respondent określał rodzaj prowadzonej **działalności gospodarczej**. Posłużono się tu podziałem zbliżonym do stosowanego w sprawozdawczości statystycznej wg działów gospodarki narodowej i otrzymano następujące wyniki (w procentach): Budownictwo 3,7; Rolnictwo 5,6; Transport 7,5; Handel 17,8; Administracja 12,1; Usługi turystyczne 7,5; Bankowość i ubezpieczenia 7,5; Przemysł chemiczny 2,8; Przemysł ciężki 11,2; Elektronika, łączność 2,8; Inny 20,6 (wydruk 1).

Tabela 2. Liczba zatrudnionych w dziale informatyki

Liczba zatrudnionych w informatyce	Liczba zakładów	Odsetek
0– 3	62	57,9
4– 10	19	17,8
11– 50	24	22,4
51–100	2	1,9
Razem	107	100,0

Kolejnymi zagadnieniami były: **liczba zatrudnionych w informatyce** oraz **liczba zatrudnionych informatyków – twórców systemów** (pytania 3 oraz 4). Zatrudnienie w dziale informatyki w ankietowanych jednostkach wahało się od kilku do 100 osób (dokładny rozkład – tabela 2). Średnio w informatyce zatrudnionych było 10,7 osób (mediana 3 osoby). Trzy czwarte jednostek zatrudniało w informatyce 10 i mniej osób. Liczba zatrudnionych twórców systemów dla blisko 80% ankietowanych nie przekraczała 4 (wydruk 2). Średnio badane jednostki zatrudniały 4,4 twórców, ale mediana wynosiła 1 a dominanta 0, co świadczy o silnej skośności rozkładu. Analiza zależności między liczbą zatrudnionych ogółem, a liczbą zatrudnionych informatyków wykazała, iż wskaźnik ten wynosił:

- w 38 zakładach od 0 do 10,
- w 24 zakładach od 10 do 50,
- w 14 zakładach od 50 do 100,
- w 6 zakładach od 100 do 200,
- w 9 zakładach od 200 do 300,
- w 7 zakładach od 300 do 500,
- w 9 zakładach od 500 i więcej.

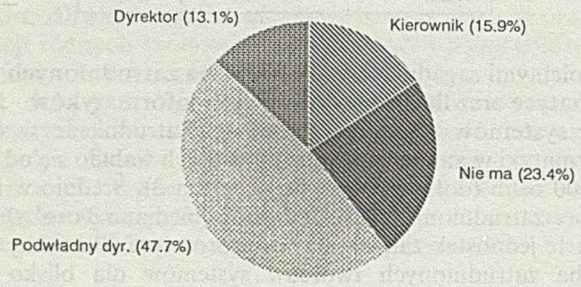


Wydruk 2. Rozkład zatrudnionych – twórców systemów

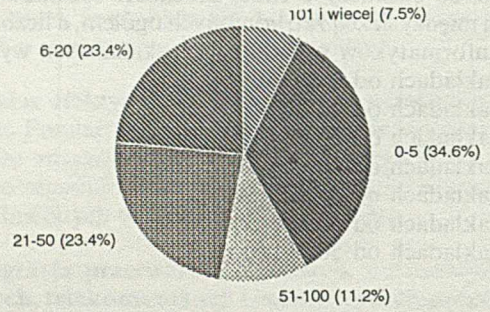
Średnio przypada 137 pracowników na jednego pracownika informatyki. Jednakże trudno na tej podstawie wyciągnąć dalej idące wnioski ze względu na duże zróżnicowanie w próbie.

W kwestii **wykształcenia osób tworzących systemy informatyczne** (pytanie 5) okazało się, iż na 107 ankietowanych przedsiębiorstw w 62 byli zatrudnieni w dziale informatyki twórcy systemów (57,9%). W 20 ankietowanych jednostkach (32,2% wszystkich przedsiębiorstw zatrudniających twórców systemów) w dziale informatyki były zatrudnione osoby z wykształceniem średnim. W tych przedsiębiorstwach stanowiły one 22,6% wszystkich zatrudnionych – twórców systemów (mediana 16,6%). Osoby z wykształceniem pomaturalnym były zatrudnione również w 20 jednostkach (32,2%), gdzie stanowiły one 39,0% wszystkich zatrudnionych (Me = 26,8). Osoby z wykształceniem niepełnym wyższym stwierdzono w siedmiu przedsiębiorstwach (11,3%). W przedsiębiorstwach tych stanowiły one jednak aż 65,6% wszystkich zatrudnionych twórców systemów (Me = 50,0%). Pracownicy z wykształceniem wyższym byli zatrudnieni w 56 przedsiębiorstwach (90,3%) i stanowili tam 82,1% wszystkich zatrudnionych (Me = 97,5%).

Kolejną analizowaną kwestią było **miejsce działu informatyki w strukturze firmy** (pytanie 6). Okazało się, że w blisko jednej czwartej badanych przedsiębiorstw w ogóle nie ma wyodrębnionego działu informatyki. Jednakże dominuje zasada nadawania w przedsiębiorstwach i instytucjach wysokiego priorytetu informatyce, bowiem ponad 60% działów informatyki podlega dyrektorowi naczelnemu lub jego bezpośredniemu podwładnemu. Dokładna struktura została przedstawiona na wydruku 3.



Wydruk 3. Zwierzchnik działu informatyki



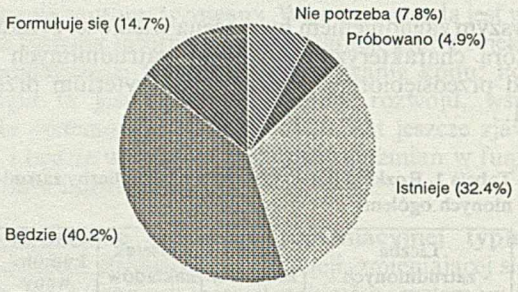
Wydruk 4. Rozkład przedsiębiorstw według liczby posiadanych terminali ogółem

Pewnym syntetycznym miernikiem stopnia informatyzacji firmy jest **liczba komputerów i terminali** (pytanie 7). Rozkład liczby terminali przedstawia wydruk 4. Średnia liczba osób przypadających na jeden terminal wynosi w badanych jednostkach 33,675 (Me = 14). Miernik w grupach według wielkości zatrudnienia ogółem kształtował się następująco:

Liczba zatrudnionych	Liczba osób na 1 terminal
0– 10	2,966
10– 50	7,825
50– 100	6,779
100– 500	30,369
500–1000	59,353
1000–5000	66,585
5000 i więcej	71,358
Ogółem	**33,675**

A zatem im większy zakład pracy, tym słabsze relatywnie wyposażenie w terminale, w tym komputery osobiste.

Miarą pewnej dojrzałości informatyzacji firmy jest dysponowanie przez nią odpowiednim planem informatyzacji, przygotowanym przy użyciu metody IBM-owskiej (metody *Bussiness System Planning*) lub podejścia równorzędnego (pytanie 8). Okazało się tu, że polscy informatycy przykładają do tego zagadnienia duże znaczenie. Taki bowiem plan istnieje w zakładzie pracy (35,1%), jest w trakcie formułowania (16,0%) lub próbowano go tworzyć (5,3%), a dalsze 40,2% jest przekonanych, iż taki plan będzie potrzebny. Jedynie 8,2% uważa plan informatyzacji za zbędny (wydruk 5).



Wydruk 5. Plan informatyzacji

Zakłady pracy na ogół nie stosują **formalnej procedury pomiaru poziomu satysfakcji użytkowników** z usług i produktów działu informatyki (pytanie 9), poza przypadkiem kiedy zwierzchnikiem działu jest dyrektor naczelny – 35,7% (tabela 3).

Tabela 3. Stosowanie formalnej procedury pomiaru satysfakcji użytkowników

Czy istnieje formalna procedura pomiaru satysfakcji	Zwierzchnik działu informatyki				Razem
	Nie ma działu	Dyrektor naczelny	Podwładny dyrektora	Kierownik n. szczebla	
Tak	3 27,3 12,5 2,8	5 45,5 35,7 4,7	1 9,1 2,0 9	2 18,2 11,8 1,9	11 10,4
Nie	21 22,1 87,5 19,8	9 9,5 64,3 8,5	50 52,6 98,0 47,2	15 15,8 88,2 14,2	95 89,6
Razem	24 22,6	14 13,2	51 48,1	17 16,0	106 100,0

Pierwszy wiersz oznacza liczebność, drugi udziały procentowe dla rozkładu wg zwierzchnika działu, trzeci udziały procentowe dla rozkładu wg pytania „Czy istnieje formalna procedura pomiaru satysfakcji", ostatni to rozkład wg obu cech łącznie

Kolejna grupa pytań dotyczyła oceny **kosztów tworzenia systemów informatycznych, użytkowania pakietów zastosowań, kosztów eksploatacji** (pytania 10–12). Największym przedmiotem zainteresowania jest analiza kosztów eksploatacji systemów – 53,8% oraz kosztów tworzenia systemów 47,5% (dominuje ocena metodą ilościową), znacznie mniejszą wagę przykłada się do kosztów użytkowania pakietów zainteresowań (tabela 4).

Tabela 4. Ocena kosztów tworzenia systemów informatycznych

Wyszczególnienie	Tak		Nie		Razem	
	n_i	%	n_i	%	n_i	%
Analiza kosztów tworzenia systemów informatycznych	48	47,5	53	52,5	101	100,0
Analiza kosztów tworzenia pakietów	35	33,3	70	66,7	105	100,0
Analiza kosztów eksploatacji istniejących systemów	57	53,8	49	46,2	106	100,0

Interesujące wyniki dała **analiza podejść (metodyk) stosowanych w tworzeniu systemów informatycznych w zakładzie pracy** (pytanie 13). Okazuje się, że najczęściej stosowanym podejściem jest zakup i adaptacja gotowego pakietu (83% wszystkich zakładów). Sytuacja ta stwarza więc wciąż duże szanse dla firm informatycznych. Spośród zaawansowanych paradygmatów tworzenia systemów informatycznych bardzo wysokim stopniem stosowalności charakteryzuje się **prototypowanie systemów** (42,0%). Procent użytkowania przez zakłady pracy poszczególnych podejść przedstawia się następująco:

- Prototypowanie systemów – 42,0%,
- Podejście tradycyjne – 30%,
- Podejście strukturalne – 26%
- Podejście zautomatyzowane – 20%,
- Podejście oparte na cyklu życia systemu – 17%
- Inne podejścia – 13%.

W badaniu podjęto próbę analizy skorelowania stosowania wyżej wymienionych metod z wielkością zakładu pracy, umieszczeniem działu informatyki w strukturze organizacyjnej oraz sformułowaniem planu informatyzacji. Zależności te przedstawiono w tabelach 5, 6 i 7. Zawarte w tabelach liczby oznaczają średni procent czasu przeznaczony na stosowanie każdego z tych podejść dla firm stosujących dane podejście (nie sumują się zatem one do 100% ani w wierszach, ani w kolumnach).

Tabela 5. Podejścia stosowane w tworzeniu systemów a liczba zatrudnionych w dziale informatyki

Podejście	Ogółem	Liczba zatrudnionych			
		0–3	4–10	11–50	51–100
Klasyczne	28,4	16,8	24,2	39,1	30,0
	32	9	9	13	1
Cykl życia	21,2	15,4	20,6	21,8	50,0
	18	5	5	7	1
Strukturalne	23,0	12,1	16,8	34,0	–
	27	8	7	12	–
Zautomatyzowane	23,4	–	19,6	30,6	–
	21	3	5	13	–
Prototypowanie	27,1	27,5	21,6	26,1	80,0
	43	18	8	16	1
Zakup	57,3	72,3	57,3	25,5	20,0
	66	34	16	15	1
Inne	17,7	24,2	14,5	12,6	20,0
	12	4	4	3	1

Pierwszy wiersz, to udział procentowy, a drugi – liczebność

Zauważono tu pewne prawidłowości. I tak ze wzrostem wielkości zakładu wzrasta procent użytkowników podejścia klasycznego, cyklu życia systemu, strukturalnego i zautomatyzowanego, natomiast spada liczba użytkowników prototypowania i zwolenników zakupu pakietów zastosowań. Zaobserwowano również, że najmniejsza liczba stosowania zaawansowanych systemów informatycznych występuje w przypadku kierowania działem informatyki przez kierownika niższego szczebla (tabela 6). Wówczas właśnie następuje tendencja do zakupów gotowego oprogramowania. Podobne małe zainteresowanie tymi podejściami wykazuje grupa firm uznających za zbędny plan informatyzacji (tabela 7). Większe zainteresowanie zaawansowanymi metodami tworzenia systemów informatycznych wykazują respondenci, u których taki plan sformułowano, formułuje się lub uważa się, że będzie on potrzebny.

Tabela 6. Podejścia stosowane w tworzeniu systemów a zwierzchnik działu informatyki

Podejście	Ogółem	Zwierzchnik		
		Dyrektor naczelny	Podwładny dyrektora	Kierownik n. szczebla
Klasyczne	28,6	39,7	28,8	10,4
	30	8	17	5
Cykl życia	21,5	13,4	34,1	14,4
	16	5	6	5
Strukturalne	22,9	30,2	35,7	3,4
	25	7	13	5
Zautomatyzowane	24,7	22,2	30,5	1,7
	20	10	9	4
Prototypowanie	28,2	25,7	34,1	9,4
	41	9	26	5
Zakup	59,5	25,7	63,4	69,1
	66	9	42	15
Inne	19,4	9,2	15,0	29,4
	11	4	2	5

Pierwszy wiersz to udział procentowy, a drugi liczebność

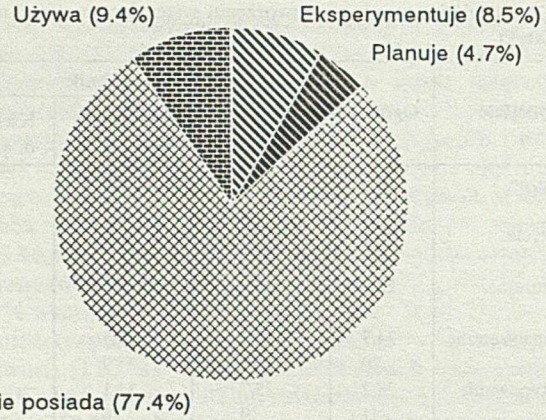
Tabela 7. Podejścia stosowane w tworzeniu systemów a istnienie planu informatyzacji

Podejście	Ogółem	Plan informatyzacji				
		Istnienie	Formułuje się	Próbowano tworzyć	Będzie potrzebny	Nie potrzeba
Klasyczne	27,5	37,8	18,3	16,0	28,0	17,0
	33	12	6	3	9	2
Cykl życia	20,1	18,5	40,0	–	20,1	14,5
	19	4	1	–	8	2
Strukturalne	22,2	21,9	26,6	–	26,2	13,0
	28	10	3	–	9	3
Zautomatyzowane	22,3	31,7	36,2	–	7,6	–
	22	8	4	–	6	–
Prototypowanie	27,0	30,6	41,4	–	23,1	12,0
	45	14	7	–	16	4
Zakup	66,6	64,1	64,2	49,6	72,5	57,5
	89	25	13	5	36	7
Inne	22,2	35,8	–	19,2	19,5	–
	14	5	–	5	2	–

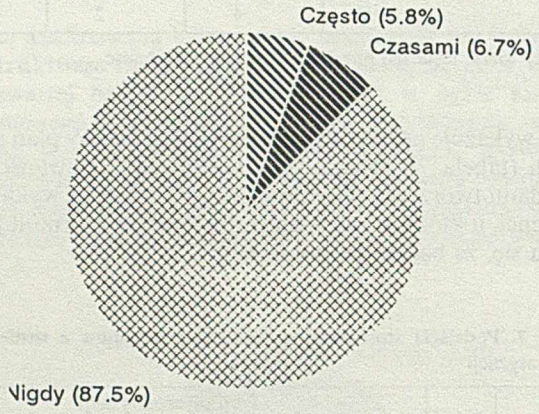
Pierwszy wiersz to udział procentowy a drugi liczebność

Jak wykazały odpowiedzi na pytania od 14 do 16, poziom stosowania najbardziej zaawansowanych narzędzi tworzenia systemów informatycznych, a więc CASE (*Computer Aided Systems Engineering*), JAD (*Joint Application Design*) oraz reengineering jest nadal niski i waha się od 3,8 do 9,4%. Procent ten jest wyższy w przypadku przedsiębiorstw zatrudniających informatyków – twórców systemów i wynosi odpowiednio od 6,5 do 16,1%. Szczegółowe wyniki przedstawiono na wydrukach 6, 7 i 8.

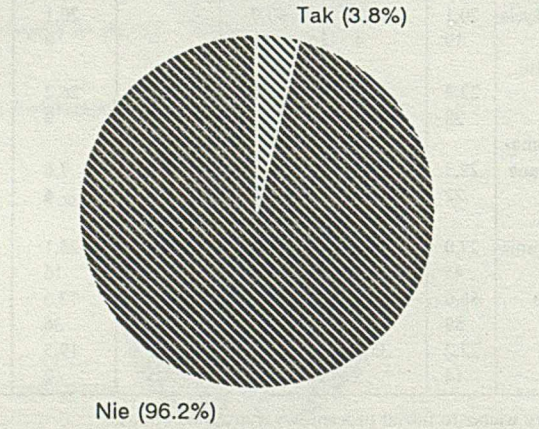
Jak wykazały odpowiedzi na pytanie 18, większość zakładów pracy (ok. 2/3) nie ma centrum informacyjnego, czyli formalnej grupy lub jednostki organizacyjnej obsadzonej głównie przez informatyków, pomagającej użytkownikom samodzielnie tworzyć oprogramowanie z zakresu ich działalności.



Wydruk 6. Użytkowanie narzędzi CASE



Wydruk 7. Użytkowanie Joint Application Design (JAD)



Wydruk 8. Wykorzystywanie metody „Reengineering"

W odpowiedzi na pytanie o **cenę działu informatyki** (pytanie 19), okazało się (tabela 8), że pracownicy działu przeciętnie wyżej oceniają jego funkcjonowanie niż użytkownicy i dyrekcja. Najwyższe oceny uzyskiwały działy informatyki, którymi kieruje kierownik niższego szczebla.

Pytania 20 i 21 dotyczyły **polityki zakupu sprzętu** przez ankietowane zakłady pracy. Wyniki zbiorcze przedstawione zostały w tabelach 9–12. Obserwuje się tendencje do zakupu sprzętu zgodnego z założeniami w przypadu mniejszych zakładów pracy. Kryteria cenowe wiążą się bardziej z wielkimi zakładami pracy. Na pewno niepośrednią rolę odgrywa tu wyższa świadomość profesjonalna i lepsze rozeznanie sprzętowe dużych zespołów informatycznych. Z kolei, jeśli chodzi o motywy zakupu komputerów osobistych, dominuje przekonanie, że ta klasa komputerów nie rozwiązuje problemów, ale są one łatwo dostępne. Małe firmy (tabela 11) na ogół (80%) uważają, iż komputery osobiste rozwiązują ich problemy.

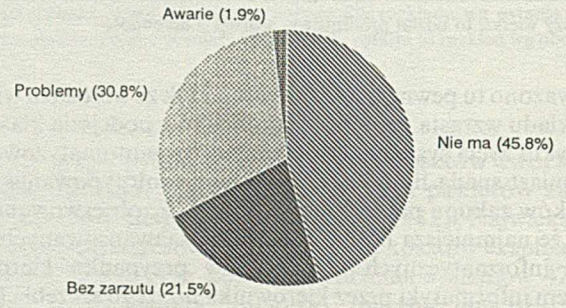
Tabela 8. Ocena działu informatyki

Wyszczególnienie	Średnia ranga	Odchyl. stand.	Liczebność
Pracownicy			
Ogółem	7,5065	2,3431	77
Nie ma działu	5,8750	3,3568	8
Dyrektor naczelny	6,6364	1,9117	11
Podwładny Dyr. naczelnego	7,6522	2,2825	46
Kierownik niższego szczebla	8,8333	1,1934	12
Użytkownicy			
Ogółem	6,7692	2,1618	78
Nie ma działu	5,3000	2,5841	10
Dyrektor naczelny	6,8182	1,7215	11
Podwładny Dyr. naczelnego	6,8696	2,0397	46
Kierownik niższego szczebla	7,6364	2,2923	11
Zwierzchnik			
Ogółem	6,6282	2,4181	78
Nie ma działu	5,4000	2,8752	10
Dyrektor naczelny	6,4167	2,1088	12
Podwładny Dyr. naczelnego	6,6957	2,4843	46
Kierownik niższego szczebla	7,8000	1,4757	10

Tabela 9. Polityka zakupu sprzętu wg wielkości zakładu

Polityka zakupu	Wielkość zatrudnienia							Razem
	0–10	10–50	50–100	100–500	500–1000	1000–5000	5000 i więcej	
Zakup tańszego sprzętu	1 2,2 20,0 1,0	4 8,7 36,4 3,9	7 15,2 33,3 6,9	14 30,4 48,3 13,7	7 15,2 46,7 6,9	8 17,4 50,0 7,8	5 10,9 100,0 4,9	46 45,1
Zakup sprzętu zgodny z założeniami	4 7,1 80,0 3,9	7 12,5 63,6 6,9	14 25,0 66,7 13,7	15 26,8 51,7 14,7	8 14,3 53,3 7,8	8 14,3 50,0 7,8		56 54,9
Razem	5 4,9	11 10,8	21 20,6	29 28,4	15 14,7	16 15,7	5 4,9	102 100,0

Pierwszy wiersz oznacza liczebność, drugi – udziały procentowe dla rozkładu wg wielkości zatrudnienia, trzeci – wg polityki zakupu, a ostatni, to rozkładu wg obu cech łącznie



Wydruk 9. Sieć lokalna

Tabela 10. Polityka zakupu sprzętu a organizacja działu informatyki

Polityka zakupu sprzętu	Zwierzchnik działu informatyki				Razem
	Nie ma działu	Dyrektor naczelny	Podwładny dyrektora	Kierownik niższego szczebla	
Zakup tańszego sprzętu	14 30,4 56,0 13,7	5 10,9 38,5 4,9	22 47,8 44,9 21,6	5 10,9 33,3 4,9	46 45,1
Zakup sprzętu zgodny z założeniami	11 19,6 44,0 10,8	8 4,3 61,5 7,8	27 48,2 55,1 26,5	10 17,9 66,7 9,8	56 54,9
Razem	25 24,5	13 12,7	49 48,0	15 14,7	102 100,0

Pierwszy wiersz oznacza liczebność, drugi – udziały procentowe dla rozkładu wg zwierzchnika działu informatyki, trzeci – wg polityki zakupu, a ostatni, to rozkład wg obu cech łącznie

Tabela 11. Motywy zakupu komputerów osobistych a wielkość zakładu

Motywy zakupu	Wielkość zatrudnienia							Razem
	0–10	10–50	50–100	100–500	500–1000	1000–5000	5000 i więcej	
Rozwiązują wszystkie problemy	4 10,5 80,0 4,1	5 13,2 41,7 5,1	9 23,7 47,4 9,2	11 28,9 40,7 11,2	6 15,8 46,2 6,1	3 7,9 17,6 3,1		38 38,8
Nie rozwiązują problemów, ale są łatwo dostępne	1 2,0 20,0 1,0	7 14,0 58,3 7,1	9 18,0 47,4 9,2	13 26,0 48,1 13,3	7 14,0 53,8 7,1	9 18,0 52,9 9,2	4 8,0 80,0 4,1	50 51,0
Nie rozwiązują problemów, ale mają tanie oprogramowanie			1 11,1 5,3 1,0	3 33,3 11,1 3,1		4 44,4 23,5 4,1	1 11,1 20,0 1,0	9 9,2
Razem	5 5,1	12 12,2	19 19,4	27 27,6	13 13,3	17 17,3	5 5,1	98 100,0

Pierwszy wiersz oznacza liczebność, drugi – udziały procentowe dla rozkładu wg wielkości zatrudnienia, trzeci wg motywów zakupu PC, a ostatni, to rozkład wg obu cech łącznie

Tabela 12. Motywy zakupu komputerów osobistych a organizacja działu informatyki

Polityka zakupu sprzętu	Zwierzchnik działu informatyki				Razem
	Nie ma	Dyrektor	Podwładny dyrektora	Kierownik niskiego szczebla	
Rozwiązują wszystkie problemy	9 23,7 40,9 9,2	8 21,1 66,7 8,2	19 50,0 38,8 19,4	2 5,3 13,3 2,0	38 38,8
Nie rozwiązują problemów, ale są łatwo dostępne	11 22,0 50,0 11,2	22 4,0 16,7 2,0	27 54,0 55,1 27,6	10 20,0 66,7 10,2	50 51,0
Nie rozwiązują problemów, ale mają tanie oprogramowanie	2 22,2 9,1 2,0	2 22,2 16,7 2,0	3 33,3 6,1 3,1	2 22,2 13,3 2,0	9 9,2
Razem	22 22,4	12 12,2	49 50,0	15 15,3	98 100,0

Pierwszy wiersz oznacza liczebność, drugi – udziały procentowe dla rozkładu wg zwierzchnika działu informatyki, trzeci – wg motywów zakupu PC, a ostatni, to rozkład wg obu cech łącznie

Tabela 13. Wyposażenie w sieć lokalną

Wyszczególnienie	Liczebność	%
Nie ma	49	45,8
Działa bez zarzutu	23	21,5
Są sporadyczne problemy	33	30,8
Częste awarie	2	1,9
Razem	107	100,0

Jak potwierdziły odpowiedzi na pytanie 22, większość dużych i średnich zakładów pracy ma sieć lokalną (tabela 13), przy czym 21,5% (wydruk 9) ankietowanych uważa, że działa ona bez zarzutu.

Kolejnym zagadnieniem była **ocena szkoleń dostępnych na rynku** (pytanie 23). Sumaryczne wyniki przedstawiają się następująco:

Sprzęt – liczba szkoleń	5,79
Sprzęt – jakość szkoleń	4,85
Oprogramowanie – liczba szkoleń	6,42
Oprogramowanie – jakość szkoleń	5,12

Ankietowani wyżej oceniali szkolenia z zakresu oprogramowania niż sprzętu, ponadto wyraźnie widać, że ankietowani nisko oceniają jakość oferowanych szkoleń (wyższą rangę ma liczba). Obliczono także średnie rangi w dwóch grupach: przedsiębiorstwa, które tworzą oprogramowanie oraz te, które je w całości kupują. Okazało się, że jednostki, które tylko kupują oprogramowanie zdecydowanie niżej oceniają jakość oferowanych szkoleń (szczególnie w zakresie sprzętu) od jednostek, które również samodzielnie tworzą oprogramowanie. Są oni częściej uczestnikami różnorodnych szkoleń. Dokładne wyniki przedstawiono w tabeli 14.

Tabela 14. Ocena szkoleń

Wyszczególnienie	Ogółem	Zakupujący	Tworzący
Sprzęt – liczba	5,79	5,52	5,91
Sprzęt – jakość	4,85	4,50	5,00
Oprogramowanie – liczba	6,42	6,38	6,43
Oprogramowanie – jakość	5,12	4,96	5,18

Bardzo zdecydowane zależności wystąpiły przy analizie **zaangażowania dyrekcji w rozwój systemów i służb informatycznych** (pytania 24 oraz 25). Otóż rozwój ten był najbardziej inspirowany i wspomagany w tych firmach, gdzie zwierzchnikiem działu informatyki był dyrektor, następnie w kolejności bezpośredni podwładny dyrektora naczelnego, a najmniej – kierownik niższego szczebla. Najtrudniejsze zadanie mają informatycy w zakładach, gdzie nie wyodrębniono takiego działu.

Jeżeli chodzi o **kryteria zakupu nowych systemów informatycznych** (pytanie 26), to w praktyce polskiej informatyki ich znaczenie jest następujące:

- jakość systemów – 94,5%
- opinia innych użytkowników – 85,9%
- cena – 79,3%
- licencja producenta – 73,6%
- okres gwarancyjny – 50,5%
- opinia o producencie – 49,5%

Wyniki te świadczą o pewnych ograniczeniach w możliwościach zakupu sprzętu, np. dobra opinia o producencie nie oznacza możliwości zakupu tego sprzętu, mimo że jakość systemów jest najważniejszym kryterium zakupu.

Ostatnie pytanie w pierwszej części ankiety było otwarte i dotyczyło oceny przyczyn niepowodzeń we współpracy z partnerami zewnętrznymi w dziedzinie systemów informatycznych. Najczęściej powtarzające się odpowiedzi to:

- wielka różnorodność dostępnego na rynku sprzętu komputerowego i oprogramowania systemowego powodująca, że opracowane systemy informatyczne, funkcjonujące na różnych platformach sprzętowych i w różnych systemach operacyjnych, są ze sobą niekompatybilne,
- brak efektywnego rozpoznania oprogramowania narzędziowego wynikający z jego różnorodności,
- tradycyjny sposób zarządzania przedsiębiorstwami, który ogranicza efektywność stosowania informatyki,
- brak jednoznacznie wypracowanych i zdefiniowanych polskich norm i standardów w zakresie projektowania, programowania i dokumentowania systemów informatycznych,
- brak aktów prawnych jednoznacznie określających i chroniących prawa autorskie twórców systemów informatycznych,
- brak sieci teleinformatycznej,
- ograniczenia finansowe,
- niska kultura informatyczna.

LITERATURA

[1] Dekleva S., Zupančič J.: An Empirical Investigation of Key Issues in Information System Management; w [3] str. 95–104

[2] Hornville G., Jowell R.: Survey Research Practice. Heinemann, London 1978

[3] Niederman F., Brancheau J. C., Wetherbe J. C.: Information Systems Management Issues in the 1990. Management Information Systems Research Center, MISRC-WP-91-08, University of Minnesota, Minneapolis, November 1990

[4] Sudman S.: Applied Sampling. Academic Press, NY 1976

[5] Wrycza S. (ed): Information Systems Developers Workbench. Preceedings of the 3rd International Conference, Sopot 1992.

meditronik SPÓŁKA z o.o.

UMC UNITED MICROELECTRONICS CORPORATION

HEWLETT PACKARD COMPONENTS

BOURNS

- CZĘŚCI ELEKTRONICZNE
- KOMPUTERY PS/1,PS/2
- DRUKARKI HP
- INSTALACJE SIECI KOMPUTEROWYCH

Partnerzy handlowi: ANALOG DEVICES, ITT, MOTOROLA, SAMSUNG, TELEFUNKEN i inni

PRZEDSTAWICIELSTWO

- UKŁADY PAMIĘCI
- UKŁADY KOMPUTEROWE
- UKŁADY KOMUNIKACYJNE I KOMERCYJNE

- TRANSOPTORY
- WSKAŹNIKI ŚWIETLNE
- WYŚWIETLACZE LED
- PRODUKTY KODÓW KRESKOWYCH
- KONTROLERY I CZUJNIKI RUCHU
- TECHNIKA ŚWIATŁOWODOWA
- ELEMENTY W.CZ. I MIKROFALOWE
- PODZESPOŁY DO MONTAŻU POWIERZCHNIOWEGO (SMD)

DYSTRYBUCJA

- POTENCJOMETRY TRIMPOT
- HYBRYDY REZYSTOROWE
- REZYSTORY SUBMINIATUROWE
- BEZPIECZNIKI MULTIFUSE
- POTENCJOMETRY PRECYZYJNE
- POTENCJOMETRY PANELI CZOŁOWYCH I KODERY
- CEWKI I TRANSFORMATORY
- CZUJNIKI CIŚNIENIA, POŁOŻENIA I PRZYŚPIESZENIA

PMI

DYSTRYBUCJA

IBM Business Partner

meditronik sp. z o.o.

00-194 Warszawa,ul. Dzika 4
tel. (02) 6352263, 6352264
fax (02) 6352195, tlx 816075

SO/575/91

Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Polska Akademia Nauk
Instytut Informatyki Teoretycznej i Stosowanej

Digital Equipment Polska

uruchamiają od 15 października roku akademickiego 1993/94
2-semestralne zaoczne
Studium Podyplomowe
na temat

SIECI KOMPUTEROWE i SYSTEMY MIKROKOMPUTEROWE

Studium obejmuje następujące przedmioty :

1. Podstawy budowy i oprogramowania sieci
2. Sieci rozlegle
3. Sieci lokalne
4. Sieci przemysłowe
5. Budowa i oprogramowanie systemów mikrokomputerowych
6. Projektowanie przemysłowych sieci komputerowych
7. Systemy rozproszone
8. Zastosowanie sieci komputerowych
9. Wykład monograficzny

Zajęcia obejmują wykłady, ćwiczenia i laboratoria. Część zajęć będzie prowadzona przez zaproszonych zagranicznych wykładowców.

Ukończenie studium daje podstawę do ubiegania się o stopień specjalizacji zawodowej zgodnie z Uchwałą Rady Ministrów Nr 66 z dnia 6 VI 1983r.

Na studium mogą być przyjęte osoby, które posiadają dyplom magistra inżyniera, inżyniera, magistra Wydziałów Informatyki, Automatyki, Elektroniki, Łączności oraz kirunków zbliżonych.

Kandydaci na Studium winni złożyć w dziekanacie:

- podanie o przyjęcie
- życiorys
- odpis dyplomu ukończenia studiów wyższych (lub kserokopie)
- deklaracje pokrycia kosztów Studium

Podania przyjmuje i informacji udziela Dziekanat Wydziału Automatyki, Elektronki i Informatyki, ul. W. Pstrowskiego 16

W PRZEDPŁACIE DUŻO TANIEJ

W II kwartale br. ukaże się na rynku drugie już wydanie **„Słownika skrótów angielskich stosowanych w elektronice, informatyce i telekomunikacji"**, najobszerniejsze, zawierające ok. 10 000 haseł, nowoczesne i bardzo poszukiwane, o czym świadczy fakt, że pierwsze wydanie rozeszło się błyskawicznie! Słownik można otrzymać pocztą pod wskazany adres, wpłacając preferencyjną, taką samą jak przy pierwszym wydaniu (pomimo inflacji!), kwotę 50 000 zł na konto:

PBK S.A. III O/Warszawa nr 370015-1573-139-11
Wydawnictwo SIGMA-NOT Sp. z o.o.,
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004

Zamawiając drugą edycję słownika w przedpłacie, ma się gwarancję otrzymania go po cenie ulgowej, bez wysokiej marży sieci handlowej i kosztów wysyłki.

Uwaga!

INFORMATYKĘ już można zaprenumerować także na poczcie.

Szczegóły na II stronie okładki

ALEKSANDER LAURENTOWSKI
KRZYSZTOF ZIELIŃSKI
Instytut Informatyki AGH
Kraków

SR – obiektowo zorientowany język programowania rozproszonego

Celem artykułu jest przybliżenie języka SR – jednego z rozwijanych obecnie narzędzi programowania równoległego i rozproszonego w lokalnych sieciach komputerowych. Jako język stosunkowo niskiego poziomu SR umożliwia efektywne programowanie. Ponadto SR stanowi środowisko do tworzenia systemów w pełni obiektowo zorientowanych.

Spośród wielu istniejących języków programowania rozproszonego SR wyróżnia się przede wszystkim **prostotą** zdefiniowanej składni oraz **wielością** dostępnych mechanizmów synchronizacji i komunikacji. Jego autorzy postawili sobie za cel zgromadzenie w jednym przejrzystym języku konstrukcji, za którymi kryją się prawie wszystkie dobrze znane z teorii systemów operacyjnych i programowania równoległego metody komunikacji i synchronizacji, takie jak m.in. zdalne wywołanie procedury (RPC), dynamiczne kreowanie procesów, przekazywanie sygnałów, rendez-vous czy semafory. Niektóre z nich można odnaleźć również w Adzie, CSP, Parallel, C, Lindzie czy notacji rejonów krytycznych i monitorów, ale w SR są one dostępne wszystkie w jednym środowisku. Są to w większości mechanizmy stosunkowo niskiego poziomu, jednak dzięki dużym możliwościom ich łączenia programista może tworzyć nawet bardzo złożone konstrukcje, elastycznie dostosowując rozwiązania do potrzeb problemu. Dodatkowo starano się nie stracić na efektywności i sile owych mechanizmów. Wydaje się, iż zamysł ten autorom powiódł się w znacznej mierze.

SR (w pełnym brzmieniu: *Synchronizing Resources*) jest projektem powstałym na początku lat 80. i rozwijanym do dziś na University of Arizona w Tucson (USA). Grupą badawczą SR kierują tam Gregory R. Andrews i Ronald A. Olsson [1]. Przy projektowaniu SR oparto się przede wszystkim na Moduli, Pascalu i DP (*Distributed Processes*). Najwcześniejsza wersja języka, zwana dziś SR_0 przewidywała mechanizmy asynchronicznego przekazywania wiadomości (ang. *asynchronous message passing*) oraz rendez-vous. Implementacja tego ostatniego była na owe czasy unikalna: proces obsługujący rendez-vous (czekający) mógł ustalać kolejność obsługi wywołań na podstawie wartości przekazywanych parametrów [2]. W miarę upływu czasu koncepcję i składnię języka udoskonalano i rozszerzano [3], a w 1988 r. udostępniono (jako oprogramowanie *public domain*) bardzo bogatą implementację środowiska SR w wersji 1.1 [4]. Artykuł niniejszy oparto głównie na doświadczeniach z eksploatacji tej właśnie wersji. Obecnie jest osiągalna wersja 2.0 SR z marca 1922 r. szczegółowo omówiona w [6][1]. Poniżej zostanie przedstawiona ogólna struktura języka i charakter dostępnych w nim mechanizmów, a w części końcowej artykułu – technika projektowania programów w SR oraz właściwości aktualnej implementacji języka.

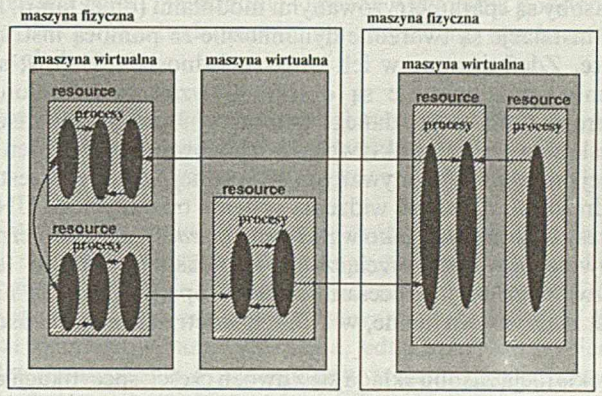
Pracę wykonano w ramach grantu KBN nr 8.0077.91.01

[1] Pakiet SR w wersji 2.0 jest dostępny via anonymous ftp z *cs. arizona.edu.*

Ogólna charakterystyka języka

Model obliczenia

Napisany w SR program może wykonywać się jednocześnie w wielu przestrzeniach adresowych przyporządkowanych do zbioru komputerów, stanowiących węzły sieciowe. Hierarchia środowiska programowego jest czterostopniowa (rys.). Najwyższy poziom, to maszyny fizyczne – jedno- lub wieloprocesorowe komputery, połączone siecią lokalną. W przestrzeniach adresowych maszyn fizycznych rezydują **maszyny wirtualne,** będące dzielonymi obszarami pamięci dla umiejscowionych w ich wnętrzu **zasobów** (ang. *resources*) oraz **globaliów** (ang. *globals*).



Model obliczenia w SR

Zasoby są zasadniczymi jednostkami języka i de facto obiektami w rozumieniu klasyfikacji systemów obiektowo zorientowanych, ponieważ zawierają one struktury danych (określające stan obiektu) oraz operacje, dzięki którym są możliwe obliczenia i komunikacja. Językowe pojęcie zasobu umożliwia tworzenie obiektów, zarówno w modelu pasywnym (udostępniającym zespół operacji, których wykonania żąda się głównie z zewnątrz), jak i aktywnym, w którym operacje należące do danego obiektu są wywoływane i wykonują się wewnątrz niego (mówimy wtedy o **procesach implementujących obiekt**); często stosowane modele mieszane są dostępne w sposób naturalny. Wszelkie jednostki (ang. *entities*), takie jak zdefiniowane wewnątrz zasobu typy, stałe czy operacje, mogą pozostać

jego własnością lub być udostępnione na zewnątrz dla grupy zasobów „zainteresowanych" ich wykorzystaniem. Globalia służą do specyfikacji jednostek globalnych – umożliwiają dzielenie między zasobami tych typów i stałych, które zgłoszą takie życzenie.

Procesy wewnątrz zasobu mogą bez przeszkód komunikować się ze sobą w ramach wspólnej przestrzeni adresowej, jaką jest dla nich zasób oraz z procesami znajdującymi się w innych zasobach tej samej lub odrębnej maszyny wirtualnej, w szczególności umiejscowionej na oddalonym fizycznym węźle sieci. Mechanizmy komunikacyjne są niezależne od położenia maszyn wirtualnych i transparentne na poziomie języka. Na przykład, asynchroniczne przekazanie wiadomości między procesami rezydującymi w jednej maszynie wirtualnej ma tę samą składnię i znaczenie co taki sam mechanizm, działający między dwoma różnymi maszynami.

Zastosowania języka sięgają od najprostszych modeli obliczeń sekwencyjnych, realizowanych na pojedynczym procesorze, przez obliczenia z pamięcią dzieloną na architekturach wieloprocesorowych, aż po systemy rozproszone, działające na zbiorze maszyn wirtualnych rozlokowanych na sieci stacji roboczych.

Podstawowe konstrukcje języka

Program w SR składa się z trzech (oddzielnie kompilowanych) części:
- globaliów (ang. *globals*),
- specyfikacji zasobów (ang. *resource specifications*),
- ciał zasobów (ang. *resource bodies*).

Globalia zawierają deklaracje stałych i typów dzielonych między zasobami. Ich deklaracja ma następującą formę:

```
global identyfikator
    deklaracje stałych i typów
end identyfikator
```

Zasoby są sparametryzowanymi modułami (obiektami), których instalacje są tworzone dynamicznie za pomocą instrukcji **create.** Zdefiniowane w ich wnętrzu jednostki, tj. typy, stałe zmienne oraz operacje są obiektami wzajemnie dzielonymi w obrębie (ang. *scope*) danego zasobu. W sensie języka zasoby, określone wyżej jako aktywne, są realizowane przez jeden lub więcej procesów wykonywanych w tej samej dzielonej przestrzeni adresowej. Z punktu widzenia systemu operacyjnego UNIX, procesy wykonują się jako wątki (ang. *threads*), a współbieżność zapewnia się przez przełączanie kontekstu procesora (ang. *context switching*). Procesami systemu operacyjnego są natomiast maszyny wirtualne, w których wnętrzu rezydują zasoby.

Deklaracja zasobu składa się z dwóch części: specyfikacji oraz ciała (podział ten może być pominięty, jeśli nie zachodzi potrzeba odrębnej kompilacji). Specyfikacja zasobu ma następującą składnię:

```
resource identyfikator
    import lista identyfikatorów zasobów
    deklaracje operacji, stałych i typów eksportowanych na zewnątrz
end identyfikator
```

Specyfikacja wyszczególnia wszystkie elementy, które zasób udostępnia na zewnątrz (eksportuje) – mogą być nimi operacje, stałe oraz typy. Mechanizm importu umożliwia korzystanie z tych elementów. Udostępnia się je przez wyliczenie po dyrektywie **import** odpowiednich zasobów. Każdą z operacji deklaruje się podając po słowie kluczowym **op** jej nazwę (identyfikator) oraz typy argumentów i zwracanej wartości. Definicje operacji, opisujące sposób ich realizacji, znajdują się w ciele (ang. *body*) zasobu.

Ciało zasobu zawiera: procesy implementujące zasób, deklaracje elementów dzielonych między tymi procesami oraz kod początkowy i końcowy (opcjonalnie):

```
body identyfikator(lista parametrów formalnych)
    import lista identyfikatorów
    deklaracje lokalnych obiektów dzielonych
    procesy
    initial blok kodu początkowego end
    final blok kodu końcowego end
end identyfikator
```

Ciało zasobu dziedziczy wszystkie obiekty importowane lub zadeklarowane w uprzednio skompilowanej specyfikacji. Składnia języka dopuszcza opcjonalność większości wyżej opisanych deklaracji, a ponadto mogą one pojawiać się w dowolnej kolejności. Obowiązuje jednak zasada, że obiekt (element) musi zostać zadeklarowany przed pierwszym odwołaniem do niego.

Oto jeden z najprostszych programów, jakie można napisać w SR:

```
resource hello()
  process writing
    write("Hello world!")
  end
end
```

Mechanizmy programowania obiektowo zorientowanego

Spójrzmy na zasób SR jako obiekt: jego *metody*, to operacje zawarte w specyfikacjach – udostępnione na zewnątrz umożliwiają dokonywanie działań na obiekcie; jego *stan* opisują struktury danych zadeklarowane wewnątrz (w ciele) zasobu. Środowiska oparte na obiektach i zawierające przestrzeń klas, tj. obiektowo zorientowane (ang. *object-oriented environments*) mają kilka specyficznych własności. Jedną z nich jest dziedziczność w hierarchii klas. Rodziny obiektów w SR mogą być tworzone za pomocą mechanizmu rozszerzania (ang. *extension*) zasobów bardziej abstrakcyjnych, rozumianych jako klasy. Mechanizm ten uzyskuje się przez dodanie do specyfikacji aktualnie definiowanego zasobu dyrektywy **extend,** po której następować powinny nazwy zasobów, których wszystkie zadeklarowane w odpowiednich specyfikacjach elementy mają być dziedziczone, np.

```
resource Figure
  op calc_area() returns area : real
end Figure
```

jest abstrakcyjnym typem – modelem figury geometrycznej, pomyślanym pod kątem obliczania jej pola. Model ten możemy dla własnych potrzeb ukonkretnić z użyciem dyrektywy **extend:**

```
resource Circle
  extend Figure
body Circle(radius : int)
  proc calc_area() returns area
    area := 3.1416*radius*radius
  end
end Circle
```

Poszerzmy jeszcze naszą rodzinę obiektów:

```
resource Square
  extend Figure
body Square(side : int)
  proc calc_area() returns area
    area := side*side
  end
end Square

resource Complement
  import Circle, Square
  op calc_area_complement() returns real

body Complement( size : int )
  var cc : cap Circle
  var cs : cap Square
```

```
    initial
        cc := create Circle(size)
        cs := create Square(2*size)
    end

    proc calc_area_complement() returns diff
        diff := cs.calc_area() - cc.calc_area()
        destroy(cs)
        destroy(cc)
    end

end Complement
```

Funkcja (metoda) *calc_area_complement* ( ) służy do obliczania różnicy pól: kwadratu o boku *a* oraz koła o średnicy *a*. Kwalifikator **cap** oznacza *capability*, czyli wskaźnik na obiekt, będący tu zasobem. Można tworzyć i posługiwać się tego typu wskaźnikami także w odniesieniu do maszyn wirtualnych, a nawet samych operacji. Zasoby można tworzyć (i niszczyć) dynamicznie za pomocą instrukcji **create** (i **destroy**) w lokalnej przestrzeni adresowej lub na zdalnych maszynach wirtualnych. Umożliwia to kwalifikator **on,** dodawany do linii polecenia **create.**
I tak:

```
var my_machine : cap vm
var my_resource : cap Complement
var difference : real

my_machine := create vm() on "tulip"
my_resource := create Complement(888) on my_machine
difference := my_resource.calc_area_complement()
```

oblicza różnicę pól między kwadratem o boku równym 888 oraz kołem o identycznej średnicy przez utworzenie zasobu *Complement* z odpowiednim parametrem aktualnym i wykonanie jego jedynej operacji (metody). Zasób ten jest tworzony na maszynie wirtualnej *my_machine,* uprzednio umiejscowionej na komputerze o nazwie „tulip", będącym węzłem sieci lokalnej. Kwalifikator **cap** oznacza tu wskaźnik (ang. *capability*) na maszynę wirtualną, o czym informuje predefiniowana stała **vm.** Warto zwrócić uwagę, iż ten sam mechanizm tworzenia obiektów – w sensie składniowym i pojęciowym – stosuje się symetrycznie do maszyn wirtualnych i zasobów choć, jak już wspomniano, ich reprezentacja w systemie operacyjnym jest różna. Jest to efekt realizacji założenia projektowego SR: najprostsza możliwa składnia przy mnogości dostępnych mechanizmów.

Operacje i mechanizmy komunikacji

Tak jak zasoby są modelami obiektów, operacje są modelami akcji wykonywanych na obiektach. Deklaracje operacji mogą pojawiać się w specyfikacjach zasobów, samych zasobach oraz wewnątrz procesów. Zadeklarowane operacje mogą być wywoływane za pomocą instrukcji **call** lub **send.** W zależności od sposobu wywoływania zmienia się mechanizm synchronizacji i komunikacji między procesami (patrz tabela).

Sposób wywołania	Operacja	Efekt
call	**proc**	wywołanie procedury (w szczególności zdalne lub rekursywne)
call	**in**	rendez-vous
send	**proc**	tworzenie procesu
send	**in**	przesłanie sygnału

Instrukcje wywołania

W SR wyróżnia się dwie instrukcje wywołania:
- **call** operacja (*lista argumentów*),
- **send** operacja (*lista argumentów*),

gdzie „operacja" jest jedną z operacji zasobu aktualnie definiowanego, reprezentującą operację polem wskaźnika (ang. *capability*) na określony zasób lub wskaźnikiem na samą operację. Argumenty mogą być przekazywane przez wartość (kwalifikator **val** – domyślny), zmienną (kwalifikator **var**) oraz (zwracane) jako rezultaty (**res**).

Instrukcja **call** zakończy się, gdy wywoływana operacja wykona się (lub zostanie obsłużona) i rezultaty zostaną zwrócone albo zostanie wykryty błąd lub awaria (patrz punkt „Obsługa błędów wykonania"). Wykonanie **send** kończy się bezwarunkowo. Stąd **call** jest instrukcją synchroniczną, a **send** – asynchroniczną. Domyślnie nie ma ograniczenia na sposób wywołania operacji, można jednak takie nałożyć, dodają do deklaracji operacji (w specyfikacji zasobu) odpowiednią restrykcję: {**call**} lub {**send**}.

Typy operacji

Operacja może być realizowana jako:
- **proc** albo
- jedna lub więcej instrukcji **in.**

Operacja typu **proc** jest uogólnieniem procedury. Ma ona następującą ogólną składnię:

```
proc identyfikator_operacji(lista parametrów ) returns identyfikator_rezultatu
        blok
end identyfikator_operacji
```

gdzie „identyfikator_operacji" jest identyczny z użytym w deklaracji, znajdującej się w specyfikacji zasobu. W momencie wywołania instrukcji **proc** powstaje jej instancja, której do wykonania przydzielany jest wątek (ang. *thread*). W zależności od sposobu wywołania, operacja wykonuje się jak proces lub procedura (w szczególności zdalna): **call** zawiesza proces-ojca do momentu obsłużenia lub przerwania instrukcji, **send** natomiast jest rodzajem rozwidlenia wątku (ang. *lightweight fork*), jako że proces macierzysty i instancja instrukcji **proc** wykonują się współbieżnie (jako wątki).

Innym typem operacji są instrukcje **in**, mające ogólną strukturę:

in *gałąź* [ ]...[ ] gałąź **ni**

Każda z *gałęzi* ma składnię i sposób wywołania podobny do **proc,** może być jednak uzupełniona o wyrażenia synchronizujące i kolejkujące:

```
nazwa_operacji (lista_parametrów) returns rezultat
        and wyrażenie_synchronizujące
        by wyrażenie_kolejkujące → blok
```

Instrukcja **in** zawiesza proces do momentu, gdy jedna z operacji, stanowiących gałęzie zostanie wywołana i odpowiadające jej logiczne wyrażenie synchronizujące przyjmie wartość *TRUE*. Wówczas wykonywany jest kod zawarty w *bloku*. Gdy kilka gałęzi spełnia powyższe wymagania, jedna z nich jest wybierana niedeterministycznie. Poszczególne wywołania danej operacji są obsługiwane w kolejności, w jakiej się pojawiały, co może być zmienione za pomocą **by** – najwyższy priorytet uzyskają wtedy wywołania o najmniejszej wartości arytmetycznego wyrażenia kolejkującego. Wyrażenie to może być pominięte. Wyrażenie synchronizujące również jest opcjonalne i implicite prawdziwe w przypadku pominięcia. Co istotne, oba te wyrażenia mogą odwoływać się do wartości parametrów operacji. Zależnie od sposobu wywołania (**call** lub **send**), instrukcja **in** zapewnia mechanizm synchronizacji typu rendez-vous lub przesyłania komunikatu (ang. *message send/receipt*). Łączy więc w sobie cechy zarówno instrukcji **select** języka Ada [10], jak i strzeżonej instrukcji wejścia (ang. *guarded input statement*) CSP [8]. Możliwość wpływania na mechanizm synchronizacji za pomocą argumentów wywołań znacznie wzmacnia siłę języka i może bardzo uprościć rozwiązania wielu problemów programistycznych (przynajmniej na poziomie logicznym).

W języku dostępne są pewne uproszczenia (skróty składniowe), umożliwiające zwięzły zapis mechanizmów komunikacji, np. instrukcja:

receive operacja (*zmienne*)

odpowiada instrukcji **in**, która oczekuje na wywołanie „operacji", a następnie nadaje „zmiennym" wartości parametrów aktualnych. Zgodnie z oczekiwaniami operacja ta, wywoływana przez **send,** zapewnia asynchroniczny sposób przesyłania wiadomości, natomiast w połączeniu z wywołaniem **call** – przesyłanie synchroniczne.

Przy użyciu instrukcji **send** i **receive** można w prosty sposób zaimplementować operacje semaforowe, umożliwiające synchronizację dostępu do zmiennych dzielonych wewnątrz zasobu. Przykładowo, jeśli *s* jest operacją zadeklarowaną:

op *s*( )

to instrukcje:
receive *s*( )
send *s*( )

symulują odpowiednio operacje semaforowe *P* i *V*. W istocie operacje zadeklarowane i wywoływane w ten właśnie sposób są wewnętrznie implementowane tak, jak gdyby były semaforami.

Inne mechanizmy komunikacji

SR dostarcza kilku instrukcji znacząco rozszerzających możliwości współpracy procesów i ich wzajemnego oddziaływania:

- **co** – umożliwia współbieżne wywoływanie i wykonywanie operacji:

```
co instrukcja ⟶ blok instrukcji końcowych
// ...
// instrukcja ⟶ blok instrukcji końcowych
oc
```

Poszczególne „instrukcje" są wywołaniami pojedynczych operacji przez **call** lub **send.** Po zakończeniu obsługi każdego wywołania opcjonalnie przewidziano możliwość wykonania dodatkowego *bloku instrukcji końcowych* (ang. *post-processing block*). Wykonanie instrukcji **co** kończy się, gdy zostaną obsłużone wszystkie wywołania i bloki końcowe lub gdy w jednym z tych bloków wykona się instrukcja **exit.** W tym drugim wypadku niedokończone wywołania nie zostaną przerwane, ale będą wykonywać się „w tle", równolegle z wątkiem, który wywołał **co.** Jest to interesująca właściwość, zwłaszcza jeśli wyobrazimy sobie np. zreplikowaną rozproszoną bazę danych, której wszystkie pliki muszą zostać uaktualnione, pomimo iż proces uaktualniający zakończył działanie po dokonaniu zmian w zaledwie kilku z nich. W każdej z wykonywanych równolegle instrukcji może być użyta zmienna sterująca, określająca liczbę powtórzeń tejże instrukcji, np.:

co (i: = **1 to** N) **call** *file* [*i*].*update*(values) oc

gdzie *file* jest tablicą wskaźników do zasobów plikowych;

- **return** – powoduje zakończenie wykonania najbardziej zagłębionej na danym poziomie operacji **in** lub **proc** (efekt po stronie obsługującego). W przypadku wywołania operacji przez **call** instrukcja ta również kończy działanie (efekt po stronie wywołującego);
- **reply** – działa podobnie jak **return,** ale pozwala operacji obsługującej istnieć dalej i wykonywać się, pomimo iż parametry **proc** zostały zwrócone do wywołującego (umożliwia to implementację tzw. mechanizmów *konwersacji* [5]);
- instrukcje sekwencyjne i iteracyjne: **if,** pętle **do** i **fa** (*for all*), które oparte są na strzeżonych instrukcjach Dijkstry [9].

Przykład

Omówione mechanizmy staną się zapewne jaśniejsze po przedstawieniu prostego przykładu. Proponujemy rozwiązanie klasycznego problemu czytelników i pisarzy: wielu czytelników może jednocześnie mieć dostęp do zasobu (czytać książkę), natomiast pisarze mają wyłączność na pracę z książką, tzn. gdy jeden z pisarzy pisze, nikt inny nie ma prawa wykonywać jakichkolwiek operacji na dzielonym zasobie. Proponowane rozwiązanie to jeden zasób SR, w którym procesy czytelników i pisarzy rywalizują ze sobą o dostęp do dzielonej zmiennej, przy czym synchronizację tego dostępu zapewnia następujący specjalny proces koordynujący:

```
resource book
body book()

# tu deklaracja zmiennej dzielonej pomiędzy procesami

  const NREADERS := 5
  const NWRITERS := 3
  var num_of_readers := 0
  var writer_active := false
  op go(i : int) {send}
  op start_reading(strd : int)        # parametry określają
  op end_reading(enrd : int)          # nr procesu wykonującego
  op start_writing(stwr : int)
  op end_writing(enwr : int)


  process reader(r := 1 to NREADERS)
    var i : int
    receive go(i)  # oczekiwanie na sygnał do startu
    send start_reading(r)
    # czytanie
    send end_reading(r)
  end

  process writer(w := 1 to NWRITERS)
    var i : int
    receive go(i)  # j.w. – a więc wszyscy mają na starcie równe szanse
    send start_writing(w)
    # pisanie
    send end_writing(w)
  end

  process coordinator
    var who : int

    do true →
        in start_reading(who) and not writer_active →
            write("czytelnik", who, "czyta")
            num_of_readers++

        [] end_reading(who) →
            write("czytelnik", who, "konczy czytanie")
            num_of_readers--

        [] start_writing(who) and not writer_active & num_of_readers= 0 →
                write("pisarz", who, "pisze")
                writer_active := true

            [] end_writing(who) →
            write("pisarz", who, "konczy pisanie")
            writer_active := false
        ni
    od
  end


  process startup
    co ( i := 1 to NREADERS + NWRITERS ) send go(i) oc
  end
end
```

Proces *startup* pełni funkcję startera dla „czekających w blokach" procesów-zawodników: umożliwia wszystkim sprawiedliwie równy start. Mimo iż trzy procesy pisarzy sygnalizują jednocześnie chęć dostępu do książki, proces *coordinator* realizuje zawsze tylko jedno z tych żądań.

Obsługa błędów wykonania

Systemy rozproszone przez swą złożoność bardziej niż inne są podatne na wszelkiego rodzaju błędy i awarie swoich elementów. Ponadto ze swej definicji powinny cechować się wysokim stopniem niezawodności dostarczanych usług. Stąd potrzeba

istnienia nie tylko niezawodnych narzędzi i technik tworzenia takich systemów, ale również mechanizmów wykrywania i zapobiegania skutkom nieuchronnych awarii.

W aktualnej implementacji języka SR nie są jeszcze dostępne procedury obsługi awarii sprzętu sieci, przepełnienia pamięci czy zniszczenia zasobów, niemniej autorzy zaplanowali dwa mechanizmy rozwiązywania tego typu problemów:
- *handlers* – procedury obsługi błędów, dowiązane do instrukcji i wywoływane w momencie wystąpienia błędu realizacji jednej z nich. Po wykryciu błędu instrukcja jest wykonywana wg przepisu zadanego procedurą obsługi błędu, a po jej zakończeniu sterowanie jeest przekazywane do procesu-ojca;
- instrukcja **when** umożliwia monitorowanie błędów powstających podczas komunikacji między procesami. Zapobiega np. zablokowaniu serwera podczas oczekiwania na wywołanie, które nigdy nie nastąpi na skutek awarii procesu, zasobu, maszyny wirtualnej lub procesu klienta, np.

when failed (*source*) **send** operation (*arguments*).

Technika tworzenia oprogramowania w SR

Konsekwencje założeń i właściwości języka

Jak już wspomniano, SR jest językiem stosunkowo niskiego poziomu. Fakt ten, jak również założenia o możliwości dynamicznej alokacji pamięci dzielonej na różnych poziomach hierarchii (czyli tworzenia maszyn wirtualnych i zasobów) nadają środowisku SR następujące cechy:
- Każdy program w SR jest oddzielną i samodzielną jednostką, będącą zbiorem maszyn wirtualnych wraz z hierarchiczną strukturą istniejących w ich wnętrzach zasobów. Nie istnieje (jak to występuje np. w Emeraldzie [9]) jądro systemu, świadczące usługi na rzecz pojawiających się wciąż na nowo aplikacji – są one w pełni samodzielne i nie mają możliwości komunikacji pomiędzy sobą.
- Struktura maszyn wirtualnych oraz moment ich utworzenia i ewentualnego usunięcia zasobów powinna być zaplanowana przez projektanta aplikacji na poziomie języka. Po kompilacji i uruchomieniu programu ani programista, ani użytkownik nie mogą dodać do niego żadnego obiektu, o ile taka możliwość nie została wcześniej przewidziana (np. możliwe jest utworzenie zasobu na podstawie analizy parametrów odczytanych z zewnątrz, jednak odpowiednia instrukcja warunkowa musi być „zaszyta" w kodzie programu).
- Programista powinien zatroszczyć się o właściwy „harmonogram" komunikacji między procesami i wybrać spośród mnogości oferowanych przez język mechanizmów odpowiednią reprezentację dla poszczególnych elementów tego dialogu.

Właściwości aktualnej implementacji języka

Obecnie dostępna implementacja języka działa na platformie systemu operacyjnego UNIX. Prowadzone są prace nad zupełnie samodzielnym środowiskiem, działającym wyłącznie w oparciu na sprzęcie i usługach sieciowych niskiego poziomu. Program w SR działa dotąd jako zbiór procesów UNIX-a (każdy reprezentując jedną maszynę wirtualną), wewnątrz których równoległość jest symulowana przez specjalny zbiór procedur niskiego poziomu, zwany RTS (ang. *run-time support*). Maszyny wirtualne wymieniają dane i komunikaty przez interfejs wtykowy (ang. *socket interface*). Zasadniczo w skład otoczenia programowego środowiska SR wchodzą: kompilator, linker i RTS. Kompilator produkuje kod pośredni w języku C, a następnie używa standardowego kompilatora C do produkcji kodu wykonywalnego. Wynika stąd cenna możliwość dołączania do programu skompilowanych procedur napisanych w języku C lub też standardowych bibliotek tego języka.

Dostępne są, wynikające z istoty UNIX-a, mechanizmy do współpracy z tym systemem:
- ponieważ obsługa wejścia-wyjścia jest związana w systemie UNIX z pojęciem pliku, możliwe są różnorodne operacje na plikach, takie jak tworzenie, otwieranie, zamykanie, zapis, odczyt i usuwanie;
- zdefiniowane są literały odpowiadające standardowym unixowym plikom: stdin, stdout, stderr;
- możliwy jest dostęp do parametrów, z jakimi został uruchomiony program w SR (ang. *command line parameters*).

* * *

W powyższym artykule przedstawiono pokrótce zarys właściwości SR – nowoczesnego języka programowania równoległego i rozproszonego. Starano się pokazać, iż jest on środowiskiem umożliwiającym programowanie obiektowo zorientowane oraz cechującym się wieloma zaletami, a przede wszystkim wielością dostępnych mechanizmów, ich elastycznością, prostotą, transparentnością i uniwersalnością. Cechy te sprawiają, iż język ten naszym zdaniem nadaje się znakomicie zarówno do nauki programowania równoległego i rozproszonego[2)], jak i tworzenia zaawansowanych oraz wymagających subtelnych i zarazem efektywnych metod synchronizacji i komunikacji programów tej klasy.

[2)] Język SR wraz z systemami Emerald i PVM jest używany przez studentów informatyki AGH podczas kursu systemów i języków równoległych i rozproszonych.

LITERATURA

[1] Andrews G.R.: Synchronizing Resources. ACM Trans. on Prog. Languages and Systems 3, 4 (Oct. 1981), pp. 405–430
[2] Andrews G.R.: The distributed programming language SR – mechanisms, design and implementation. Software-Practice and Experience 12,8 (Aug. 1982), pp. 719–754
[3] Andrews G.R., Olsson R.A.: The evolution of the SR language. Distributed Computing 1, 3 (July 1986), pp. 133–149
[4] Andrews G.R., Olsson R.A., Coffin M., Elshoff I., Nilsen K., Purdin T., Townsend G.: An overview of the SR language and implementation. ACM Trans. on Prog. Languages and Systems 10, 1 (Jan. 1988), pp. 51–86
[5] Andrews G.R.: Concurrent Programming: Principles and Practice. Benjamin/Cummings Publishing Company, 1991
[6] Andrews G.R., Olsson R.A.: The SR Programming Language: Concurrency in practice. Benjamin/Cummings Publishing Company, 1992
[7] Dijkstra E.W.: A Discipline of Programming. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976
[8] Hoare C.A.R.: Communicating Sequential Processes. Prentice-Halll, Inc. Englewood Cliffs, NJ, 1985
[9] Raj R.K., Tempero E.D., Levy H.M., Black A.P., Hutchinson N.C., Emerald E.Jul.: A General-Purpose Programming Language. Software – Practice and Experience 21, 1 (Jan. 1991), pp. 91–118
[10] U.S. Department of Defence: Reference Manual for the Ada Programming Language. ANSI/MILSTD-1815A, 1983.

W związku z wprowadzeniem powszechnego podatku dochodowego i koniecznością informowania właściwych Urzędów Skarbowych o wszystkich wypłatach zwracamy się do Autorów wysyłających teksty do opublikowania o podawanie następujących danych:
- **nazwisko, imiona (pierwsze i drugie),**
- **imiona: ojca i matki,**
- **miejsce i data urodzenia,**
- **numer identyfikacyjny PESEL (wpisany przez Biuro Meldunkowe do dowodu osobistego – nie mylić z numerem dowodu osobistego!),**
- **dokładny adres (miejsce zameldowania),**
- **adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora (bardzo ważne – bez tej informacji nie możemy przekazać honorarium do wypłaty!).**

TADEUSZ WITKOWSKI
Kombinat Przemysłu Narzędziowego VIS
Warszawa

Algorytm rozpoznawania problemu szeregowania zadań

Wśród metod wykorzystywanych przy rozpoznawaniu obrazów szerokie zastosowanie mają metody wykorzystujące teorię zbiorów rozmytych i przybliżonych [3, 4, 6, 7].

Wprowadzenie pojęcia zbiorów rozmytych miało na celu umożliwienie kwantyfikowania i formułowania niepełnej i niepewnej informacji. W teorii zbiorów rozmytych wykorzystuje się często określenia sumy i iloczynu zbiorów. Jeżeli np. A' i B' zbiory rozmyte oraz $f_{A'}(x)$ i $f_{B'}(x)$ przedstawiają wartości przynależności określonego elementu x do zbiorów A' i B', to przynależność $A' \cup B'$ określa się jako $f_{A' \cup B'}(x) = \max\{f_{A'}(x), f_{B'}(x)\}$, a przynależność $A' \cap B'$ – jako $f_{A' \cap B'}(x) = \min\{f_{A'(x)}, f_{B'(x)}\}$.

Za pomocą odpowiednich procedur wykorzystujących zbiory rozmyte i przybliżone można obliczyć wartości przynależności poszczególnych elementów (obiektów) do odpowiednich klas. Rozmytość wyniku klasyfikacji można usunąć przyrównując obiekt do tej klasy, której odpowiada największa wartość przynależności. Jedną z metod podejmowania decyzji jest rozpatrywanie wartości przynależności jako końcowy wynik procesu rozpoznawania.

Sformułowanie rozpoznawanego problemu

Rozpatrzmy zbiór zadań $Z = \{Z_i\}$, gdzie $i \in I = \{1, 2, ..., n\}$ oraz zbiór maszyn (procesorów) $U = \{u_k\}$, gdzie $k = 1, 2, ..., m$. Każde zadanie Z_i tworzy partię Π_i jednakowych zadań cząstkowych P_i określonego rodzaju. Jeżeli operacje procesu wytwarzania (przetwarzania) i-tego zadania oznaczymy przez $\{O_{ij}\}_{j=\xi_i}^{\kappa_i}$ wtedy dla Z_i możemy zapisać $Z_i = (\Pi_i, \{O_{ij}\}_{j=\xi_i}^{\kappa_i})$, przy czym Π_i – wielkość partii zadań cząstkowych P_i, $O_{ij} = (G_{ij}, t_{ij})$ – j-ta operacja wykonania i-tego zadania; ξ_i – numer operacji procesu, od której rozpoczyna się wykonanie i-tego zadania; κ_i – numer ostatniej operacji procesu z i-tym zadaniem; G_{ij} – zbiór maszyn do wykonania operacji O_{ij}; t_{ij} – czas wykonania operacji O_{ij} i-tego zadania, wykonywanego na określonej maszynie z grupy G_{ij}; t'_{ij} – pracochłonność przygotowania maszyny do wykonywania operacji O_{ij}; M – liczba operacji procesu, przy czym $M = \max_i \kappa_i$.

Poszukiwanym rozwiązaniem rozpatrywanego problemu jest określenie przynależności uszeregowań zadań, zidentyfikowanych przez strukturę randomizacji reguł priorytetowych [8, 9] do poszczególnych klas w zależności od wartości wybranej funkcji kryterialnej F. Uszeregowaniem jest przyporządkowanie w czasie maszyn ze zbioru U do zadań ze zbioru Z, dla którego są spełnione takie warunki, jak:

- w każdej chwili każda maszyna wykonuje co najwyżej jedno zadanie,
- zadanie Z_j jest wykonywane w określonym przedziale czasowym, w danej chwili co najwyżej przez jedną maszynę,
- wykonywane są wszystkie zadania,
- dla każdych dwóch zadań $Z_i, Z_j \in Z$, takich, że $Z_i < Z_j$ (tj. wykonywanie zadania Z_i musi być zakończone przed rozpoczęciem wykonywania zadania Z_j) przy czym $<$ oznacza tu ograniczenie kolejnościowe w czasie.

W zadaniu rozpoznawania dąży się do tego, aby skonstruować algorytm realizujący odwzorowanie $A: D \to I \cup \{i_0\}$, takie, aby miara $\beta(A, \hat{A})$, nazywana oceną jakości algorytmu rozpoznawania, była minimalna, przy czym jednoelementowy zbiór $\{i_0\}$ symbolizuje brak odpowiedzi. Definicja algorytmu $\hat{A}$ dopuszcza rozpoznanie wariantowe, to znaczy przyjmuje się sytuację, w której jako rozpoznanie akceptuje się dowolny podzbiór I. Takie rozwiązanie jest przydatne z punktu widzenia praktycznych zastosowań, ponieważ aby osiągnąć zamierzony cel wystarczy niekiedy wskazać określony podzbiór klas rozpoznawanych obiektów. Odwzorowanie A można opisać także jako $\hat{A}: D \to 2^I$, gdzie oznaczenie 2^I przedstawia zbiór wszystkich podzbiorów zbioru I. Odwzorowanie A jest realizowane jako złożenie trzech odwzorowań $A = F \cdot C \cdot \psi$ będących elementami składowymi rozpoznania, przy czym pierwsze z nich – $\psi: D \to X$ – jest nazywane recepcją, drugie – $C: X \to R^L$ – oznacza obliczanie wartości tzw. funkcji przynależności, zaś ostatnie odwzorowanie – $F: R^L \to I \cup \{i_0\}$ lub $F: R^L \to 2^I$ – oznacza proces podejmowania decyzji [7].

Recepcja i struktura przestrzeni cech rozpoznawanych obiektów

Początkowym elementem każdego algorytmu rozpoznającego jest pomiar cech wszystkich obiektów – zarówno wzorco-

Dr inż. TADEUSZ WITKOWSKI studiował na Politechnice Łódzkiej oraz w Ługańskim Instytucie Budowy Maszyn (Ukraina), który ukończył w 1976 r. W 1983 r. obronił pracę doktorską w Instytucie Cybernetyki im. W. Głuszkowa Akademii Nauk Ukrainy w Kijowie w specjalności „Zautomatyzowane systemy zarządzania i przetwarzania informacji".. Od 1969 r. pracował m.in. w ZM „Ursus" i FSO jako projektant i analityk systemów, a w latach 1984–1990 w Instytucie Administracji i Zarządzania w Warszawie. Obecnie pracuje jako specjalista ds. organizacji produkcji w Kombinacie Przemysłu Narzędziowego „VIS" w Warszawie. Jest autorem około 30 publikacji z zakresu komputerowego wspomagania decyzji w zarządzaniu przedsiębiorstwem.

wych, należących do ciągu uczącego, jak i podlegających rozpoznaniu $\psi : D \to X$.

Określenie cech prowadzi do zamiany obiektów $d \in D$ w punkty pewnej przestrzeni, przy czym X oznacza przestrzeń cech, której elementami są wektory n-elementowe $\underline{x} = \langle x_1, x_2, ..., x_n \rangle \in X$.

Szczególną uwagę należy zwrócić na konstrukcję funkcji przynależności zbioru rozmytego typu uszeregowanie zadań – od kolejności określonej wstępnie I_T do kolejności odwrotnej do niej I_{-T}.

Jak wiadomo, nawet przy $m = 3$ (liczba maszyn) analityczne rozwiązanie problemu szeregowania istnieje tylko przy określonych relacjach między czasami wykonania zadań. Dla ogólnego zagadnienia szeregowania dowiedziono, że jest ono problemem *NP*-zupełnym [1], dla których trudne lub prawie niemożliwe jest odnalezienie uogólnionych algorytmów rozwiązania o wielomianowym nakładzie obliczeń.

Jeżeli do szeregowania zadań wykorzystuje się tylko jedną lub kilka reguł typu np. FIFO[1] lub SIO[2], to okazuje się, szczególnie na średnich operacjach procesu, że otrzymane wyniki są odwrotne do oczekiwanych, a więc w ogólnym przypadku równomierność generowanych uszeregowań nie jest wymagana.

W przypadku idealnym, taka skłonność generatora z większym prawdopodobieństwem będzie generować tylko niektóre uszeregowania (permutacje) i powinna być wyrażona za pomocą odpowiedniej funkcji rozkładu. Ale i bez jawnego przedstawienia takiej funkcji jest możliwe skonstruowanie generatorów oraz ich wykorzystanie do określonych celów badawczych. Takie generatory uszeregowań nazywane są generatorami randomizacji [8].

Aby otrzymać uszeregowanie bliskie optymalnemu, większość generatorów liczb losowych wymaga znacznej liczby próbnych uszeregowań. Np. metody typu Monte Carlo są wolne od ścisłego schematu przeszukiwania, jednakże nie wykorzystują one żadnej informacji a priori. Ponadto konieczność wprowadzania przy konstruowaniu uszeregowań elementu losowości można uzasadniać także okolicznością ograniczonej liczby reguł priorytetowych. Nawet jeżeli dla każdej operacji procesu stosować określoną regułę, to i tak ogólna liczba możliwych kombinacji jest znacznie mniejsza od zbioru wszystkich możliwych uszeregowań.

Przedstawione wyżej okoliczności były głównymi przyczynami opracowywania algorytmów randomizowanych (ang. *randomizing algoritm*). Stosując randomizację można zmieniać daną kolejność numerów zadań aż do kolejności odwrotnej. Typowa randomizacja daje, ogólnie mówiąc, zestaw uszeregowań mniej lub bardziej zbliżonych do optymalnego, z których można wybrać najlepsze pod względem określonych kryteriów.

Pod randomizacją rozumiemy złożenie dwóch permutacji

$$B_j^Q \triangleq B_j^R \circ B_j$$

gdzie B_j^R jest permutacją losową generowaną metodą Monte Carlo, a B_j oznacza permutację zadań ustaloną przez regułę priorytetu [9]. Na przykład, jeżeli $B_j = (2,3,1,4)$ i $B_j^R = (4,2,1,3)$, to $B_j^Q = (4,3,2,1)$.

[1] FIFO – *First In – First Out*

[2] SIO – Shortest Imminent Operation

Przyjęto kompleksową regułę priorytetową B_j o postaci $\text{FIFO}_j \to \text{SIO}_j \to \text{SIO}_{j+1} \to ... \text{SIO}_M$

gdzie reguła FIFO zwalnia zadanie, które zostało wcześniej dostarczone do wykonania, a reguła SIO – zadanie, którego czas wykonania na wolnej maszynie jest najmniejszy. Powyższa postać reguły priorytetowej oznacza, że dla określenia priorytetu zadań na danej operacji stosuje się początkowo regułę FIFO_j, a następnie SIO_j. W przypadku niemożliwości określenia priorytetu na tej operacji za pomocą reguł FIFO_j oraz SIO_j, porównuje się czas wykonania zadań na następnych operacjach procesu przez wykorzystanie reguły SIO_{j+1} itd.

W jednym ze sposobów, przy konstrukcji permutacji losowej, wykorzystuje się odcinek jednostkowy podzielony na części, których długości na początku jego podziału tworzą postęp geometryczny z ilorazem Q, zależnym od numeru operacji j. Poszczególne części odcinka, odpowiadające kolejno wygenerowanym losowo liczbom, są usuwane, przy czym ich numery tworzą permutację B_j^R, natomiast pozostałe części odcinka są wzajemnie dosuwane (z prawej strony na lewą) itd. Wybór poszczególnych części odcinka, a więc odpowiadającej im liczby losowej, odbywa się przez trafienie w niego liczby z generatora liczb losowych o rozkładzie równomiernym. Ponieważ Q zależy od numeru operacji j, a więc mając $Q(j)$, możemy określić M konkretnych podziałów odcinka dla określenia M operacji.

Rozpatrzmy sposoby określania podziałów odcinków, wykorzystywanych w procesie randomizacji reguł priorytetowych przy szeregowaniu zadań na poszczególnych operacjach procesu produkcyjnego.

W zbiorze Σn wszystkich permutacji o długości n można wyróżnić dwie „skrajne" permutacje, z których jedna jest permutacją tożsamościową $J_T \triangleq \{1, 2, ..., n\}$, a druga ma postać $J_{-T} \triangleq \{n, n-1, ..., 1\}$. Ze względu na częściowe uszeregowanie zbioru Σn, nie możemy w sposób deterministyczny określać permutację za pomocą zmiany wartości $Q(j)$. Istnieje jednak pewna możliwość częściowego sterowania tym wyborem, a mianowicie:

- im mniejsza jest wartość ilorazu Q, tym większe istnieje prawdopodobieństwo, że zostanie wybrana permutacja $B_j^R = I_T$,
- przy $Q = 1$ wszystkie odcinki mają jednakową długość i permutacja B_j^R będzie całkowicie losowa,
- im większa jest wartość ilorazu Q, tym większe prawdopodobieństwo, że zostanie wybrana permutacja $B_j^R = I_{-T}$.

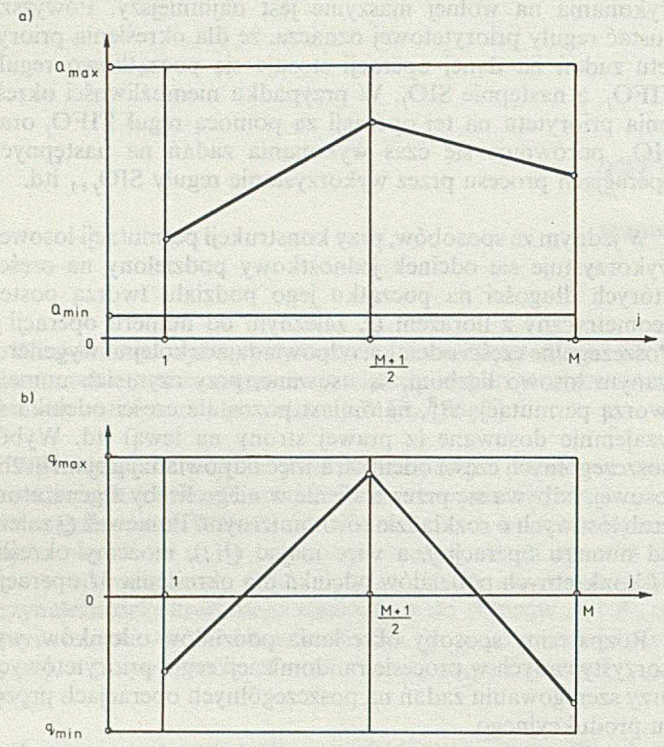
Oznaczmy przez $Q_{\min}$ i $Q_{\max}$ graniczne wartości Q, przy których z prawdopodobieństwem p (np. $p = 0{,}95$) zostanie wybrana permutacja I_T i I_{-T}. Wykorzystując procedurę podziału odcinka na części można określić równania, za pomocą których znajduje się $Q_{\min}$ i $Q_{\max}$:

$$(1 - Q_{\min})^n \Big/ \prod_{i=1}^{n} (1 - Q_{\min}^i) = p, \tag{1}$$

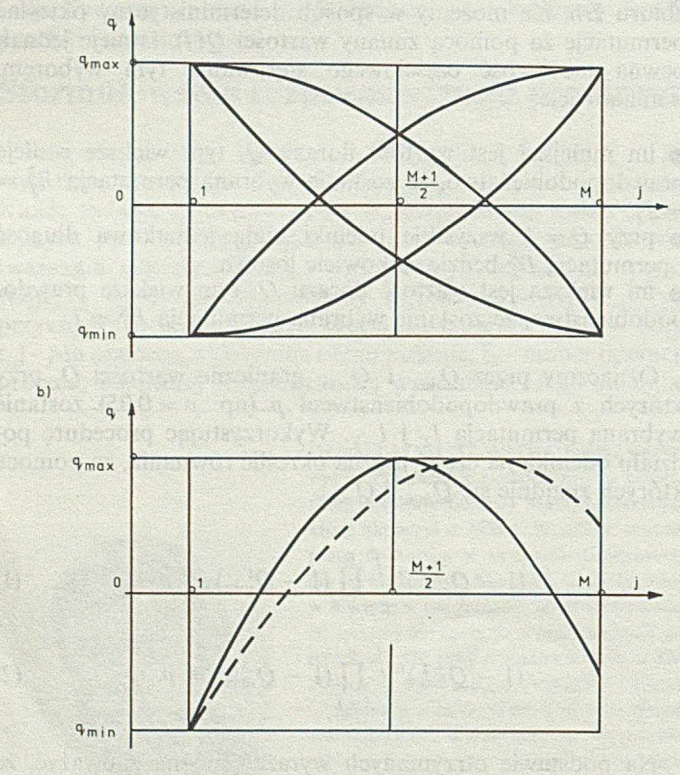
$$(1 - Q_{\max}^{-1})^n \Big/ \prod_{i=1}^{n} (1 - Q_{\max}^{-i}) = p. \tag{2}$$

Na podstawie otrzymanych wyrażeń można zauważyć, że przy jednakowych wartościach p otrzymane wartości $Q_{\min}$ i $Q_{\max}$ są wartościami wzajemnie odwrotnymi. Przy $n = 10$ i $p = 0{,}95$ na podstawie wzoru (1) otrzymujemy $Q_{\min} = 0{,}006$, a ze wzoru (2) – wartość dla $Q_{\max} = 173$.

Taki prawie deterministyczny wybór permutacji nie jest konieczny dla każdej operacji, a często wystarczy tylko, aby ilorazy $Q(j)$ przyjmowały wartości w zakresie $Q_{min} \leqslant Q(j) \leqslant Q_{max}$, przy $1 \leqslant j \leqslant M$.



Rys. 1. Graficzne przedstawienie dwóch postaci funkcji Q(j) i q(j):
a) funkcja odcinkowo-liniowa Q(j)
b) funkcja odcinkowo-liniowa w zależności logarytmicznej q(j) = lnQ(j)



Rys. 2. Graficzne przedstawienie niektórych rodzajów funkcji kwadratowej q(j) dla dwóch przypadków:
a) ekstremum paraboli leży na zewnątrz odcinka [1, M]
b) ekstremum paraboli leży wewnątrz odcinka [1, M]

Zakładamy, że permutacja zadań przy przejściu od j-tej do $(j + 1)$ operacji zmienia się stosunkowo płynnie, a także że $Q(j)$ traktowane jako funkcje j, są funkcjami ciągłymi o 1–2 przedziałach monotonności. W charakterze takich funkcji przyjmowano funkcje odcinkowo-liniowe, odcinkowo-wykładnicze i funkcje kwadratowe (rys. 1 i 2). W zależności od charakteru krzywej $Q(j)$ i sposobu jej zmiany, opracowano kilka algorytmów określania podziału odcinków.

W algorytmach tych w macierzy operacji procesu produkcyjnego $\| O_{ij} \|$, której wiersze przedstawiają wspólne operacje (wykonywane na tych samych maszynach), wybierane są trzy „bazowe" operacje: pierwsza $j = 1$, średnia j_{sR} oraz ostatnia $j = M$, gdzie $j_{sR} = M/2$ dla M parzystych oraz $(M + 1)/2$ dla M nieparzystych. Na tych operacjach w określony sposób wybierana jest trójka

$$\{Q_l(1), Q_l(j_{sR}), Q_l(M)\} = Q_l, \text{ gdzie } l \in 1, l_{max}.$$

Dla pośrednich operacji $1 < j < j_{sR}$ oraz $j_{sR} < j < M$ wartości $Q(j)$ są określane za pomocą interpolacji liniowej lub wykładniczej, m.in. w celu zmniejszenia pracochłonności obliczeń. Dla utworzenia bardziej równomiernej dyskretyzacji na operacjach bazowych $j \in \{1, j_{sR}, M\}$, zamiast wartości Q są wybierane wartości q z zakresu $[q_{min}, q_{max}]$, gdzie $q_{min} = ln\, Q_{min}$, a $q_{max} = ln\, Q_{max}$, tzn. jest wybierana uporządkowana trójka $q = \{q_l(1), q_l(j_{sR}), q_l(M)] \in Q_1 \times Q_2 \times Q_3$ spośród N^3 wariantów, przy czym Q_i jest pewną dyskretyzacją przedziału $[q_{min}, q_{max}]$, zawierającą N punktów rozdzielających odcinki. Dyskretyzację można wykonać przez podział przedziału $[q_{min}(j), q_{max}(j)]$ na równe odcinki lub za pomocą losowego wyboru N punktów dla każdej z trzech dyskretyzacji Q_i, a więc jest możliwe utworzenie $l_{max} = N^3$ wariantów.

Skonstruowana funkcja przynależności zbioru rozmytego typu „uszeregowanie zadań od kolejności określonej wstępnie do kolejności odwrotnej" jest wykorzystywana następnie do oceny algorytmów generujących uszeregowania pod względem wartości kryterialnej F. W naszym przypadku jest to minimalny sumaryczny czas wykonania wszystkich zadań.

Określanie wartości funkcji przynależności

Odwzorowanie ψ polegało na określeniu właściwości rozpoznawanego obiektu $d \in D$. W odwzorowaniu $C: X \to R^L$ chodzi o ustalenie pewnej miary podobieństwa nieznanego obiektu $d \in D$ do poszczególnych klas D_i, gdzie $i \in I$.

Na podstawie określonego wektora cech $\underline{x}$ są obliczane funkcje przynależności $C^i(\underline{x}), i = 1, 2, ..., L$. Wartości tych funkcji określają miarę przynależności nieznanego obiektu d do poszczególnych klas $D^i (i = 1, 2, ..., L)$. Przy czym dla funkcji przynależności jest formułowany postulat, aby wszystkie konkretne obiekty d^k, należące do określonej klasy o numerze i^k, były wskazywane maksymalną wartością funkcji przynależności $C^{i^k}(\underline{x}^k)$, która powinna być większa od wszystkich pozostałych wartości $C^i(\underline{x}^k)$ dla $i \neq i^k$.

Opracowana metoda oraz program komputerowy są doskonalone na zadaniach testowych. W przypadku algorytmu heurystycznego, który nie gwarantuje dokładnego rozwiązania, oraz z powodu niemożliwości sprawdzenia optymalności znalezionego rozwiązania, pozostaje testowanie algorytmu na wybranych zadaniach.

Wykorzystując dyskretyzację przedziałów $[q_{min}, q_{max}]$ na operacjach bazowych, dla pozostałych wartości konstruuje się funkcję $q(j)$, podobnie jak poprzednio funkcję $Q(j)$. W przypad-

ku jednakowych wartości q_{min} i q_{max} na operacjach bazowych, pole ograniczone funkcją $q(j)$ ma kształt prostokąta, a dla różnych wartości q_{min}, q_{max} ma kształt sześciokąta.

W procesie testowania opracowanych algorytmów dla funkcji $q(j)$ wprowadzono pojęcie sześciokąta „ramowego". Jeżeli połączyć krzywą trzy punkty odpowiadające q_{min} i q_{max} na operacjach bazowych, to wraz z dwiema rzędnymi tworzą one wspomniany sześciokąt ramowy, zawierający wszystkie możliwe w danym wariancie funkcje $q(j)$. Algorytm jest określony przez parametry, w tym również przez wartości granic dolnych q_{min} i q_{max}. Im mniejszy obszar sześciokąta ramowego, tym mniejszy jest w większości przypadków podzbiór uszeregowań znacznie różniących się wartościami funkcji kryterialnej, który może wygenerować określony algorytm.

W ten sposób, przy pozostałych jednakowych warunkach, wariant algorytmu jest nie gorszy od pozostałych, jeżeli:

- generuje uszeregowanie dające wartość funkcji kryterialnej, nie gorszą niż w pozostałych algorytmach,
- algorytm generuje liczbę uszeregowań z wartością funkcji kryterialnej nie mniej korzystną niż pozostałe algorytmy, przy czym najlepsze własne uszeregowanie dany wariant algorytmu uzyskuje podczas liczby prób nie większej niż inne warianty algorytmów.

Jeżeli wybieramy konkretny wariant algorytmu z określonymi wektorami $\bar{q}_{min}$ i $\bar{q}_{max}$, to w procesie generowania uszeregowań powstaje konieczność wyboru takiej statystyki dla algorytmu, aby z dużym prawdopodobieństwem można było określić liczbę prób $N_{dost.}$, gdzie $N_{dost.} \leqslant N^3$, przy której zostanie wygenerowane, chociażby jeden raz, najlepsze spośród uszeregowań, jakie określony algorytm może wygenerować. Przy czym najlepsze z tych uszeregowań, jakie algorytm jest w stanie wygenerować, może być w procesie generowania N^3 uszeregowań tworzone wielokrotnie.

Ponieważ ogólna liczba uszeregowań, jaką można otrzymać w procesie generowania, jest o wiele mniejsza od liczby teoretycznej, to liczba przeszukiwanych uszeregowań do otrzymania reprezentatywnej próbki jest także znacznie mniejsza od wielkości teoretycznej. Praktyka pokazała, że liczba próbnych uszeregowań generowanych określonym algorytmem równa $N^3 = 15^3 = 3375$, jest w większości przypadków próbką dostatecznie reprezentatywną.

W eksperymencie, którego schemat blokowy przedstawiono na rys. 3, wybrano przedstawione w tabl. 1 i 2, wartości q na poszczególnych operacjach bazowych.

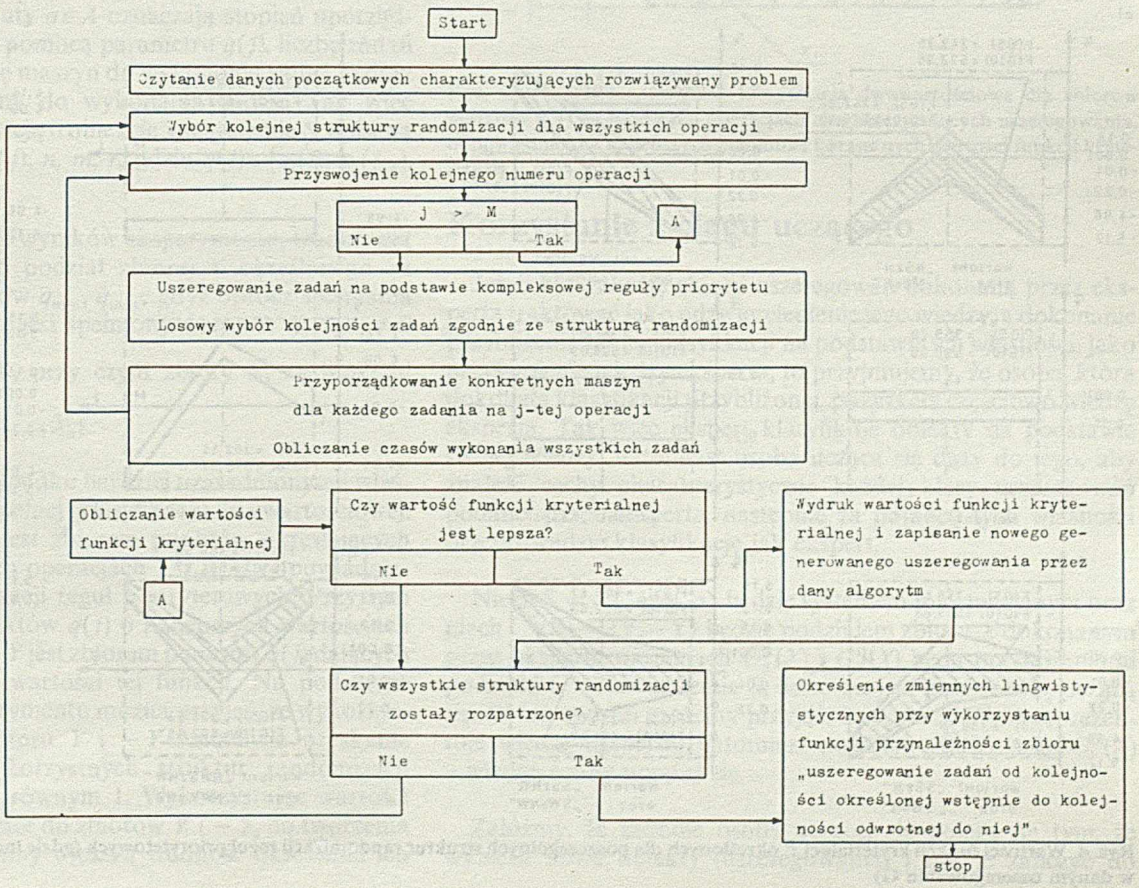
Tabela 1

Nazwa zakresu	Oznaczenie	Dolna i górna granica
Niski	N	-5,12 ; -4,96
Szeroki	Sz	-0,22 ; 0,18
Wąski	Ws	-0,01 ; 0,001
Wysoki	Wk	4,09 ; 4,14
Najwyższy	NW	5,13 ; 5,15

Tabela 2

Nazwa zakresu	Oznaczenie	Dolna i górna granica
Niski	N	-5,12 ; -4,96
Średni	S	-0,01 ; 0,001
Wysoki	W	4,09 ; 4,14
Najwyższy	NW	5,13 ; 5,15

Ze wszystkich możliwych uporządkowań kombinacji, w eksperymencie wykorzystuje się te wartości zakresów q na operacjach 1-ej, j_{sR} i M, które uwzględniają poglądy o wpływie



Rys. 3. Schemat blokowy określania stopnia prawdziwości zdania φ: „sposób randomizacji reguł priorytetowych przy szeregowaniu zadań jest najlepszy przy przyjętej funkcji kryterialnej F"

zastosowania poszczególnych reguł priorytetowych na wartości wybranej funkcji kryterialnej. Ma to na celu przede wszystkim skrócenie czasu eksperymentu symulacyjnego, w którym reguły określania struktur randomizacji określa się następująco:

IF $q(1) \in N \wedge q j_{sR}) \in N \wedge q(M) \in N$,
THEN algorytm typu $\text{FIFO}_j \rightarrow \text{SIO}_j \rightarrow \text{SIO}_{j+1}$;
IF $q(1) \in S \wedge q(j_{sR}) \in Ws \wedge q(M) \in S$,
THEN algorytm Monte Carlo;
IF $q(1) \in NW \wedge q(j_{sR}) \in NW \wedge q(M) \in NW$,
THEN algorytm typu LIFO_j[3] $\rightarrow \text{LIO}_j$[4] $\rightarrow \text{LIO}_{j+1}$; itp.

W wyniku eksperymentu uporządkowano zbiór algorytmów pod względem wartości funkcji kryterialnej F, uzyskanej dla najlepszego z uszeregowań generowanych każdym algorytmem przy jednakowej liczbie prób N^3. Kolejność ta zmierza w kierunku polepszenia wartości kryterium:

- pojedyncze uszeregowanie losowe, otrzymane metodą Monte Carlo,
- uszeregowania otrzymane za pomocą reguły $\text{LIFO}_j \rightarrow \text{LIO}_j$ i niektórych odmian reguły FIFO oraz SIO,
- najlepsze uszeregowanie z pierwszych dziesięciu uszeregowań otrzymanych metodą Monte Carlo,
- uszeregowania otrzymane za pomocą kompleksowej reguły $\text{FIFO}_j \rightarrow \text{SIO}_j \rightarrow \text{SIO}_{j+1}$,
- metoda Monte Carlo przy parametrach $q_{\min} = 0{,}01$ i $q_{\max} = 0{,}001$ na wszystkich operacjach bazowych,
- randomizowane reguły priorytetów $\text{FIFO}_j \rightarrow \text{SIO}_j \rightarrow \text{SIO}_{j+1}$ z sześciokątem ramowym typu $NWsN$ lub $NSzN$.

Ilustrację graficzną parametrów struktur randomizacji reguł priorytetowych wraz z wartościami funkcji kryterialnej dla uszeregowań z liczbą zadań 5 oraz 10 przedstawiono na rys. 4.

[3] LIFO – *Last in – First on*
[4] LIO – *Longest Imminent Operation*

Podejmowanie decyzji

Odwzorowanie $F: R^L \rightarrow I \cup \{i_0\}$ służy do ustalenia ostatecznej decyzji, tj. definitywnego rozpoznania lub jego braku. Jako zasadę podejmowania decyzji przyjęto regułę majoryzacyjną: $\bigwedge_{x \in X} [[F(C^1(x)\, C^2(x), \ldots, C^L(x)) = i] \equiv \bigwedge_{k \in I, k \neq i} [C^k(x) < C^i(x)]]$.

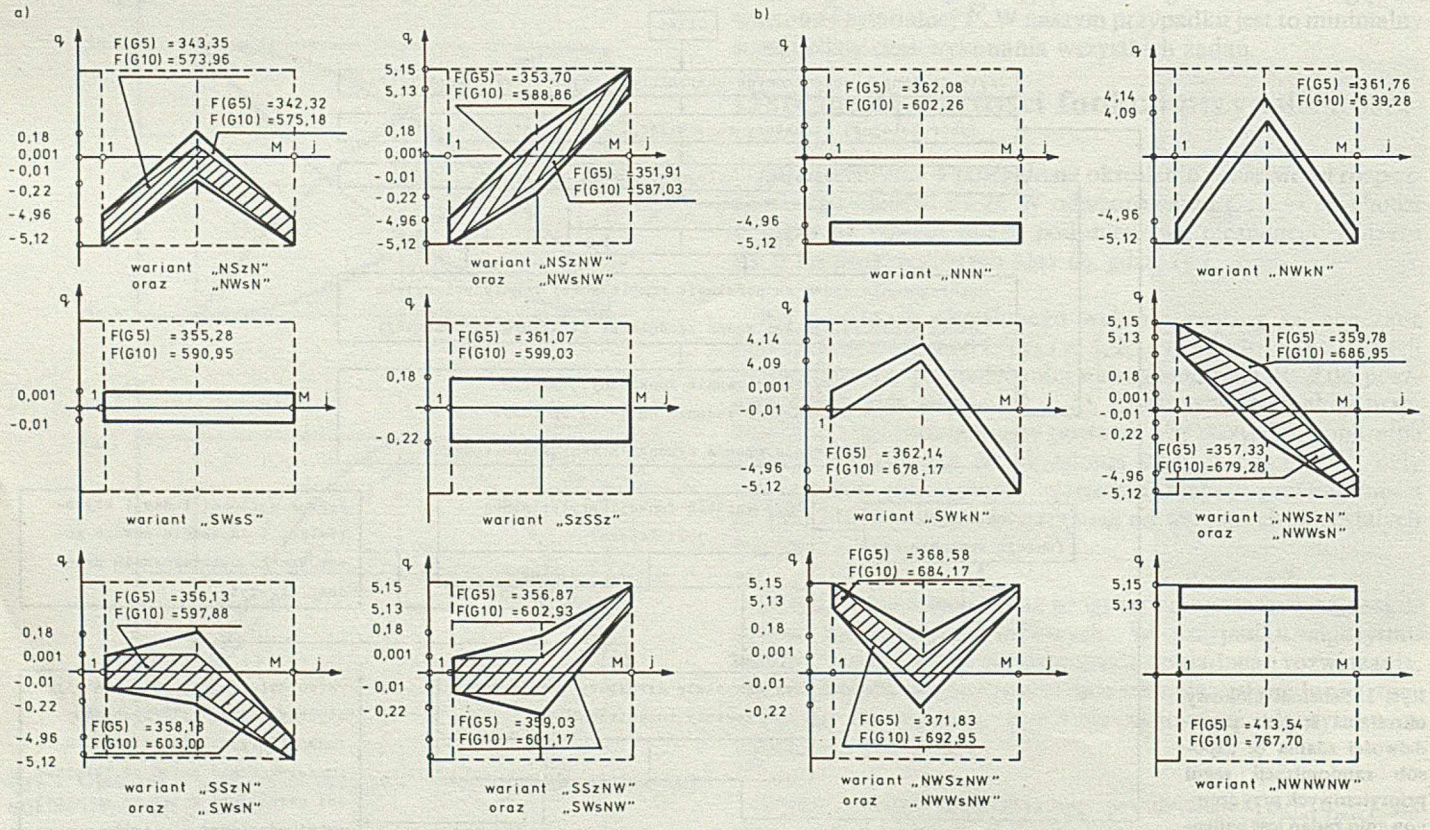
Tak więc podejmowana jest decyzja o przynależności obiektu $d \in D$ opisywanego wektorem cech $\underline{x} \in D$ do tej klasy $i \in I$, dla której wartość funkcji przynależności $C^i(\underline{x})$ jest maksymalna.

Załóżmy, że mamy określić najlepsze uszeregowania zadań do wykonania na podstawie przyjętego kryterium wyboru F. Niech będzie dane zdanie φ o postaci „struktura randomizacji $q(1)$, $q(j_{sR})$, $q(M)$ reguł priorytetowych przy szeregowaniu zadań jest najlepsza przy przyjętym kryterium F". W zależności od parametrów sterujących q temu zdaniu można przypisać odpowiednie rozmyte wartości prawdy [2, 5].

Wprowadźmy określenie funkcji przynależności zbioru rozmytego μ_q typu „uszeregowanie zadań od kolejności określonej wstępnie J_T do kolejności odwrotnej do niej J_{-T}", którego zmienne lingwinistyczne mają postać: „zgodne z J_T", „bardzo bliskie do J_T", „bliskie do J_T", „dalekie do J_T", „niezgodne z J_T i niezgodne z J_{-T}", „dalekie do J_{-T}", „bliskie do J_{-T}", „bardzo bliskie do J_{-T}", „zgodne z J_{-T}",... .

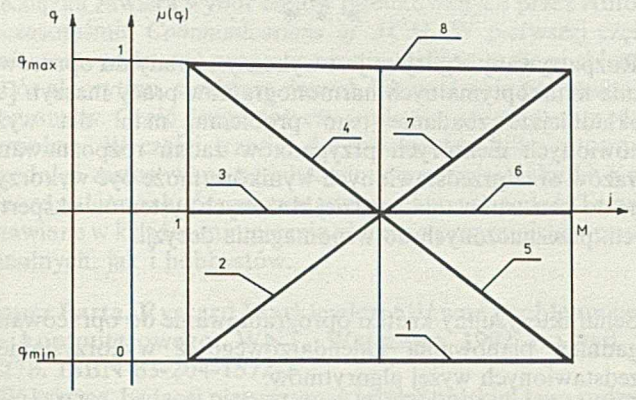
Jeżeli przy wyborze wariantów randomizacji reguł priorytetowych (przy określonej strukturze danych rzeczywistych, takich jak czas t_{ij} wykonania operacji, liczba maszyn itp.) przyjąć wartości q przedstawione na rys. 5, tj.:

1. $q_{\min}$, $\bigwedge_{j \in [1\ M]}$; 2. od $q_{\min}$ do q_{sR}, $\bigwedge_{j \in [1,\ M/2]}$; 3. q_{sR}, $\bigwedge_{j \in [1,\ M/2]}$; 4. od $q_{\max}$ do q_{sR}, $\bigwedge_{j \in [1,\ M/2]}$; 5. od q_{sR} do $q_{\min}$, $\bigwedge_{j \in [M/2,\ M]}$; 6. q_{sR}, $\bigwedge_{j \in [M/2,\ M]}$; 7. od q_{sR} do $q_{\max}$, $\bigwedge_{j \in [M/2,\ M]}$;



Rys. 4. Wartości funkcji kryterialnej F określonych dla poszczególnych struktur randomizacji reguł priorytetowych (gdzie indeks (Gn) oznacza liczbę zadań w danym uszeregowaniu G)

8. $q_{max}, \bigwedge_{j\in[1,M]}$, to wtedy rozmyte wartości prawdy dla zdania φ przyjmują następującą postać:
2 i 5 – bardzo prawdziwe,
2 i 7 oraz 3 i 6 – prawdziwe,
1, 3 i 5 oraz 3 i 7 – dość prawdziwe,
4 i 5 – mało prawdziwe,
4 i 7 – nieprawdziwe,
8 – bardzo nieprawdziwe.



Rys. 5. Warianty struktury randomizacji reguł priorytetowych w funkcji q(j) oraz rozmyte wartości prawdy dla zdania φ

Spróbujmy przeprowadzić klasyfikację wielowartościową obiektów (uszeregowań) przypisując je na podstawie parametrów $q(j)$ do jednej z grup zbiorów przestrzeni dwuwymiarowej określonej za pomocą wektorów $\bar{q}_{min}$, $\bar{q}_{max}$.

Rozpatrzmy teraz nasz problem jako system informacyjny $S\langle X, A, V, \rho\rangle$, gdzie X to skończony zbiór obiektów; A – skończony zbiór atrybutów; $V = \bigcup_{a\in A} V_a$, V_a – dziedzina atrybutu $a(V_a = \{v \in V$: dla których istnieje $x \in X$, takie, że $\rho(x, a) = v\})$; ρ – funkcja całkowita, $\rho: X \times A \to V$; przy czym $\rho(x, a) \in V_a$ dla każdego $x \in X$ oraz $a \in A$ [6].

W takim zapisie atrybuty $a \in A$ oznaczają stopień uporządkowania uszeregowań za pomocą parametru $q(j)$, liczbę zadań do uszeregowania n, liczbę maszyn do wykonania zadań m oraz regułę r wyboru maszyny do wykonania zadań. Tak więc $\rho_x(a) = \rho(x, a)$ jest informacją o obiekcie x w systemie S, opisaną za pomocą atrybutów $q(j)$, n, m, r, gdzie $q(j) \in \{q_i(1), q_i(j_{sR}), q_i(M)\}$.

Jak widać na podstawie wyników eksperymentu, trudno jest dokładnie przeprowadzić podział zbioru X określonego za pomocą wartości wektorów $\bar{q}_{min}$, $\bar{q}_{max}$, gdyż oprócz spełniania przynależności $X_i \subset X$ nie jest spełniony warunek $X_i \cap X_j \neq 0$ dla $i \neq j$ oraz $\bigcup_{i=1}^{n} X_i = X$, przy czym zbiory X_i są blokami podziału $C^0(X) = \{X_1, X_2, ..., X_n\}$.

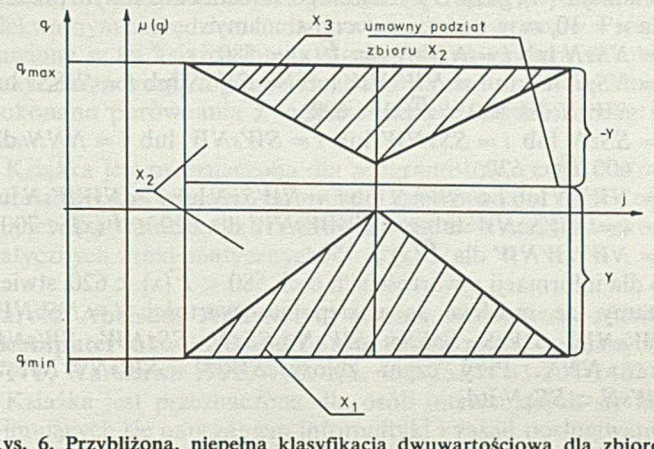
W rozpatrywanym przypadku bardziej uzasadnionym i właściwym jest przyjęcie niepełnej klasyfikacji dwuwartościowej. Przyjmijmy że $-Y$ i Y jest zbiorem punktów określających wartości $q(j)$ na bazowych operacjach 1, j_{sR}, M odpowiadających strukturze randomizacji reguł priorytetowych. Przy tym zbiór Y jest zbiorem punktów $q(j)$ o najlepszych wartościach funkcji kryterialnej, zaś $-Y$ jest zbiorem punktów $q(j)$ dających najbardziej niekorzystne wartości tej funkcji. Na podstawie przeprowadzonego eksperymentu można przyjąć, że wykorzystanie wartości $q(j)$ ze zbioru Y i $-Y$ umożliwia uzyskanie najlepszych i najmniej korzystnych struktur randomizacji z prawdopodobieństwem równym 1. Wykorzystując wartości $q(j)$ ze zbioru X, nie należące do zbiorów Y i $-Y$, do tworzenia struktur randomizacji mamy większą trudność określenia ich przynależności. Tworzymy więc trzecią klasę obiektów niezidentyfikowanych do końca. Znaczy to, że zbiór obiektów X dzieli się na trzy klasy X_1, X_2, X_3 i otrzymujemy w ten sposób klasyfikację trójwartościową, w której $Y \subset X_1 \cup X_2$, $-Y \subset X_2 \cup X_3$ oraz $X_1 \subset Y$, $X_3 \subset -Y$. Zbiór X_2 zawiera więc wszystkie elementy, o których nie potrafimy orzec, czy należą one do Y czy też do $-Y$.

Niech $S\langle X, A, V, \rho\rangle$ będzie systemem informacyjnym, Y podzbiorem zbioru X oraz $C^0(X) = \{X_1, X_2, X_3\}$ niech będzie podziałem spełniającym warunki: $Y \subset X_1 \cup X_2$, $-Y \subset X_2 \cup X_3$ oraz $X_1 \subset Y$ i $X_3 \subset -Y$.

Wtedy najlepszym przybliżeniem dolnym zbioru Y w systemie S jest $\underline{S}X_1$, tj. największy zbiór opisywalny w systemie S zawarty w X_1, zaś przybliżeniem górnym zbioru Y jest $\overline{S}(X_1 \cup X_2)$, tj. najmniejszy zbiór opisywalny w systemie S zawarty w $(X_1 \cup X_2)$.

Podobnie dla zbioru $-Y$, przybliżeniem dolnym jest $\underline{S}X_3$, a przybliżeniem górnym $\overline{S}(X_1 \cup X_3)$.

Tak więc podział dwuwartościowy $C^0(X) = \{Y, -Y\}$, $Y \subset X$ możemy w danej sytuacji przybliżyć od dołu parą zbiorów $\underline{S}X_1$, $\underline{S}X_3$, zaś od góry parą zbiorów $\overline{S}\{X_1 \cup X_2\}$, $\overline{S}\{X_2 \cup X_3\}$ (rys. 6).



Rys. 6. Przybliżona, niepełna klasyfikacja dwuwartościowa dla zbiorów punktów q(j) na bazowych operacjach charakteryzujących uszeregowania, odpowiednio dla najlepszych i najmniej korzystnych wartości funkcji kryterialnej

Korzystanie z ciągu uczącego

Jeżeli klasyfikację zbioru uszeregowań dokonaną przez eksperta traktować jako odzwierciedlenie jego wiedzy, a dokonanie przybliżenia takiej klasyfikacji na podstawie ich własności, jako przyswojenie wiedzy eksperta, to przyjmujemy, że osoba, która dokonała klasyfikacji przybliżonej, pozyskała częściowo wiedzę eksperta. Tak więc ekspert klasyfikuje obiekty na podstawie swojej wiedzy, natomiast osoba ucząca się dąży do tego, aby znaleźć cechy charakterystyczne każdej klasy uszeregowań podanej przez eksperta, następnie za pomocą tych własności przeprowadzić klasyfikację jak ekspert.

Niech $S = \langle X, A, V, \rho\rangle$ będzie systemem informacyjnym oraz niech $C^0(X) = \{Y, -Y\}$ będzie podziałem zbioru X dokonanym przez eksperta oraz niech $\overline{C}_s^0(X)$ i $\underline{C}_s^0(X)$ będą przybliżeniami podziału $C^0(X)$ w systemie S, dokonanymi przez osobę uczącą się. W ten sposób możemy przyjąć, że podział $C^0(X)$ reprezentuje wiedzę eksperta, natomiast podziały $\overline{C}_s^0(X)$ oraz $\underline{C}_s^0(X)$ – wiedzę osoby uczącej się.

Załóżmy, że zadanie osoby uczącej się polega na tym, że wybiera nowe obiekty (uszeregowania) i ma je zaliczyć do

właściwych klas na podstawie atrybutów systemu S (stopnia uszeregowania na poszczególnych operacjach). Decyzje osoby uczącej się są kontrolowane przez eksperta, który dokonuje niezależnej klasyfikacji. Dodając więc nowe uszeregowanie do zbioru X, tworzymy nowy system informacyjny S', nowy podział $C_s^0(X')$ oraz nowe jego przybliżenia $\overline{C}_s^0(X')$ i $\underline{C}_s^0(X')$.

Rozpatrzmy teraz niektóre zadania rozpoznawania [3] na podstawie reprezentacji wiedzy podanej przez eksperta wprost w formie formułowanej zależności między x oraz i, gdzie x – wektor cech, i – numer klasy.

1. Relacja $R(x, i)$ jest formułowana w taki sposób, że dane jest rozbicie X na rozłączne zbiory $\Delta x_1, ..., \Delta x_l$, takie, że $\Delta x_1 \cup \Delta x_2 \cup ... \cup \Delta x_l = X$ oraz każdemu zbiorowi $\Delta x_l (l \in 1, L)$ jest przyporządkowany zbiór $D_{il} \subset \{1, 2, ..., M^0\}$. Ponadto, jeżeli w wyniku obserwacji jest ustalony zbiór $D_x \subset X$, to zbiór możliwych klas $D_i \subset \{1, 2, ..., M^0\}$ jest następujący: $D_i = \{i: \bigvee_{x \in D_x} [(x, i) \in R(x, i)]\} = \bigcup R(x, i)$. Wówczas wyznaczenie D_i sprowadza się do następującego algorytmu: dla danego D_x należy uwzględnić wszystkie ΔX_l mające z nim punkty wspólne i wyznaczyć $D_{i'}$ jako sumę zbiorów D_{il} odpowiadających zbiorom Δx_l. Np. jeśli $X \in R^2$ i $M^0 = 18$ oraz dla poszczególnych $x = \{q(1), q(j_{sR}), q(M)\}$ w zależności od wartości funkcji kryterialnej $F(G)$, gdzie G jest uszeregowaniem określonym przez x, dla $n = 10$, $m =$ const. i $r =$ const. mamy:

$i = NSzN$ lub $i = NWsN$ dla $F(x) < 580$;

$i = NSzNW$ lub $i = NWsNW$ lub $i = SWsS$ lub $i = SzSSz$ lub $SWsN$ dla $580 \leqslant F(x) < 600$;

$i = SSzN$ lub $i = SSzNW$ lub $i = SWsNW$ lub $i = NNN$ dla $600\ F(x)\ 620$;

$i = NWkN$ lub $i = SWkN$ lub $i = NWSzN$ lub $i = NWWsN$ lub $i = NWSzNW$ lub $i = NWWsNW$ dla $620 \leqslant F(x) < 700$;

$i = NWNWNW$ dla $F(x) \geqslant 750$,

to dla informacji o wartościach cech $580 < F(x) < 620$, stwierdzamy, że możliwe są następujące wartości: $i = NSzNW$, $NWsNW$, $SWsS$, $SzSSz$, $SWsN$, $SSzN$, $SSzNW$, $SWsNW$ oraz NNN. Przy czym zbiory $SWsN = SWsN \cap NWsS$, $SWsN \subset SSzN$ itd.

2. Przypadek $R(x, i)$ w postaci ciągu uczącego $S_n = (x_1, i_1), (x_2, i_2), ..., (x_n, i_n)$. Ekspert może podawać i znając oprócz x również inne cechy (np. liczbę szeregowanych zadań), które wraz z x są określane w postaci funkcji $i = \gamma(x, a)$. W takim przypadku komputer podaje wynik rozpoznawania tylko na podstawie x, z czego wynika ewentualna możliwość błędnej klasyfikacji uszeregowań.

Dla zaobserwowanych wartości cech x algorytm generacji rozwiązania sprowadza się do algorytmu korzystania z ciągu uczącego. W szczególności może to być np. algorytm „najbliższy sąsiad", klasyfikujący x do klasy i_l, której w ciągu uczącym odpowiada wektor x_l najbliższy wektorowi x w sensie określonej normy, tj. $\bigwedge_{k \in 1,n} (\|x - x_1\| \leqslant \|x - x_k\|)$. Jeśli w ciągu uczącym powtarzają się te same wartości x_l, to jako rezultat otrzymujemy zbiór możliwych klas D_i odpowiadających wektorowi x_l w różnych elementach ciągu uczącego.

3. Relacja $R(x, i)$ ze wskaźnikiem pewności v, tj. relacja rozmyta $\mu(x, i)$. Dla danej relacji $R(x, i)$ ze wskaźnikiem pewności v oraz danego D_x otrzymujemy $D_i = \{i: \bigvee_{r \in D_x} [(x, i) \in R(x, i)]\} = \bigcup_{r \in D_x} R(x, i)$, ze wskaźnikiem pewności v, tzn. implikacja $x \in D_x \Rightarrow i \in D_i$ ma wskaźnik pewności v. W tym przypadku $\mu_i(i) = \max_{r \in D_x} \mu(x, i)$, co oznacza, że dla każdej klasy i określamy wskaźnik pewności, na podstawie którego rozpoznawany obiekt jest zaliczany do tej klasy. Ostateczna decyzja może oznaczać wybór takiej klasy i, dla której wartość $\mu_i(i)$ jest największa. Dla ciągu uczącego $S_n = (x_1, i_1), (x_2, i_2), ..., (x_n, i_n)$, z dodatkową charakterystyką niepewności w postaci $v_r = v(x_r, i_r)$ odpowiedź jest taka jak w przypadku 2, z określeniem wskaźnika pewności $v_l = (x_l, i_l)$, tj. wskaźnika pewności, że obiekt należy do klasy i_l.

* * *

Rozpatrywany algorytm jest wykorzystywany do opracowywania m.in. optymalnych harmonogramów pracy maszyn [9]. Dokładniejsze zbadanie tego problemu, m.in. dla wyżej omówionych niektórych przypadków zadań rozpoznawania obrazów oraz przedstawionych wyników, może być wykorzystane do konstruowania bardziej złożonych systemów ekspertowych, przeznaczonych do wspomagania decyzji.

Scharakteryzujmy krótko oprogramowanie do opracowania zagadnień planowania kalendarzowego z wykorzystaniem przedstawionych wyżej algorytmów.

System wspomagania decyzji jest przeznaczony do konstruowania suboptymalnych harmonogramów uruchomiania zadań dla konwencjonalnych oraz zrobotyzowanych gniazd i wydziałów z dyskretnym charakterem produkcji. System ten charakteryzuje się następującymi właściwościami:

- służy do rozwiązywania zagadnień planowania kalendarzowego różnorodnych dyskretnych procesów produkcyjnych,
- pozwala na opracowywanie harmonogramów dla części kompleksu zadań produkcyjnych w dowolnym momencie cyklu produkcyjnego (np. w przypadku awarii wyposażenia),
- umożliwia konstruowanie dowolnej liczby próbnych harmonogramów z generowaniem harmonogramów najlepszych,
- umożliwia przeprowadzenie optymalizacji harmonogramów na podstawie wielu kryteriów, takich jak: minimalizacja całkowitego czasu wykonania zadań produkcyjnych, średniego ważonego czasu przepływu części i materiałów, minimalizacji przestojów maszyn, liczby przezbrojeń oraz wymaganego w tym celu czasu itp., przy czym istnieje możliwość optymalizacji wielokryterialnej.

System działa w trybie konwersacyjnym, przy czym czas obliczeń przy wykorzystaniu procedury randomizacji jest określony zależnością $n \lg n$, (gdzie n – liczba wykonywanych zadań) i zależy liniowo od liczby operacji technologicznych. Opracowano także algorytmy o mniejszej pracochłonności obliczeń.

Oprogramowanie zostało zrealizowane w języku FORTRAN i PASCAL i może być wykorzystywane na komputerach typu IBM PC XT/AT.

LITERATURA

[1] Błażewicz J.: Złożoność obliczeniowa problemów kombinatorycznych. WNT, Warszawa, 1988, s. 194

[2] Bolc L., Borodziewicz W., Wójcik M.: Podstawy przetwarzania informacji niepewnej i niepełnej. PWN, Warszawa, 1991, s. 136

[3] Bubnicki Z.: Wstęp do systemów ekspertowych. PWN, Warszawa, 1990, s. 184

[4] Kacprzyk J.: Zbiory rozmyte w analizie systemowej. PWN, Warszawa, 1986, s. 522

[5] Klir G. J., Folger T. A.: Fuzzy sets, uncertainty and information. Prentice Hall, Englewood Cliffs, 1988, p. 355

[6] Pawlak Z.: Systemy informacyjne. Podstawy teoretyczne. WNT, Warszawa, 1983, s. 188

[7] Tadeusiewicz R., Flasiński M.: Rozpoznawanie obrazów. PWN, Warszawa, 1991, s. 216

[8] Tanajew W. S., Szkurba W. W.: Wwiedjenie w tjeoriju raspisanij. Nauka, Moskwa, 1975, s. 256

[9] Witkowski T.: Rozmyte wartości prawdy w problemie szeregowania zadań, w: Optymalizacja. Zadania, metody, algorytmy. IBS PAN-PTBOiS, Warszawa, 1991, s. 89–96.

Jon Bentley: Perełki oprogramowania. WNT, Warszawa 1992 r., wyd. 1, s. 204. ISBN 83-204-1405-9

Książka zawiera wybór esejów publikowanych przez Autora w czasopiśmie *Communications of ACM*. W pierwszej części omówiono podstawy programowania: definiowanie problemu, dobór algorytmu, strukturę danych i dowodzenie poprawności programów. Część druga dotyczy trudności związanych z efektywnością. W części trzeciej do rozwiązania kilku istotnych zagadnień zastosowano przedstawione wcześniej techniki. Każdy rozdział jest zakończony zestawem ciekawych zadań. Tematy omawiane w książce zainteresują zarówno programistów profesjonalnych, jak i hobbystów.

Janusz Barta, Ryszard Markiewicz: Główne problemy prawa komputerowego. WNT, Warszawa 1993 r., wyd. 1, s. 278. ISBN 83-204-1572-1

W książce, będącej pierwszym w języku polskim tego rodzaju opracowaniem, przedstawiono zarys prawa komputerowego i główne problemy jego stosowania. Rozważania obejmują prawo polskie (w tym również ostatnie projekty ustaw), prawo obce – zwłaszcza prawo USA i RFN, prawo EWG i prawo międzynarodowe. W tych płaszczyznach uwypuklono zasady ochrony programów komputerowych oraz umowy dotyczące tworzenia i rozpowszechniania programów. Omówiono także prawne problemy zbierania i przetwarzania informacji w związku z wykorzystaniem ich w bankach danych.

Książka jest przeznaczona dla informatyków, producentów programów i ich użytkowników, jak również dla prawników.

Arnold Rosen: PageMaker 4.0. WNT, Warszawa 1992 r., wyd. 1, s. 112, format B6. ISBN 83-204-1555-1

Książka jest przeznaczona dla wszystkich użytkowników mikrokomputerów. Zawiera opis operacji możliwych do wykonania w programie PageMaker oraz szczegółowy opis poszczególnych operacji. Na końcu zestawiono nazwy procesorów tekstu, których pliki wyjściowe mogą być importowane przez PageMarkera.

Janusz Oprzędkiewicz: Wspomaganie komputerowe w niezawodności maszyn. WNT, Warszawa 1993 r., wyd. 1, s. 224. ISBN 83-204-1529-2

Książka obejmuje problemy niezawodności maszyn, zastosowanie techniki komputerowej w opracowaniu wyników badań niezawodnościowych, symulacyjne metody badań, badania w systemie eksploatacji maszyn, budowę modeli niezawodnościowych i przykłady praktycznego wykorzystania omawianych metod. Zawiera podstawowy materiał umożliwiający inżynierom zajmującym się projektowaniem, wytwarzaniem i eksploatacją maszyn włączenie problematyki niezawodnościowej w sferę swej działalności zawodowej.

Książka jest przeznaczona dla studentów wydziałów mechanicznych i innych kierunków studiów zajmujących się projektowaniem i eksploatacją obiektów technicznych, a także dla pracowników zaplecza naukowo-badawczego przemysłu i biur projektowych.

Andrzej Wójciak: Mikroprocesory w układach przekształtnikowych. WNT, Warszawa 1992 r., wyd. 2 zmienione, s. 192. ISBN 83-204-1382-6

Książka jest poświęcona zastosowaniom mikroprocesorów do sterowania numerycznego urządzeń przemysłowych zawierających energoelektroniczne przekształtniki statyczne. W wydaniu tym znacznie ograniczono materiał dotyczący budowy podstawowego systemu mikroprocesorowego, wprowadzono natomiast informacje o mikroprocesorach sygnałowych i innych nowoczesnych elementach peryferyjnych. Znacznie zwiększono także liczbę przykładów praktycznych rozwiązań układów mikroprocesorowego sterowania urządzeń przekształtnikowych.

Książka jest przeznaczona zarówno dla studentów wydziałów elektrycznych, jak i dla inżynierów (elektryków, elektroników) zajmujących się projektowaniem i eksploatacją przemysłowych urządzeń energoelektronicznych.

Grażyna Mirkowska, Andrzej Salwicki: Logika algorytmiczna dla programistów. WNT, Warszawa 1992 r., wyd. 1 s. 300. ISBN 83-204-1296-X

Tematem tej książki jest logika algorytmiczna i jej zastosowania w analizie semantycznych własności programów, specyfikowaniu modułów programów oraz aksjomatycznym definiowaniu semantyki języków programowania. Zastosowania logiki algorytmicznej zilustrowano licznymi przykładami. Omówiono oryginalne matematyczne modele obliczeń współbieżnych. Wykazano, że klasyczna logika pierwszego rzędu nie może być adekwatnym narzędziem badania własności programów. Przedstawiona w tej książce logika algorytmiczna zawiera w sobie klasyczne rachunki logiczne i stanowi ich istotne rozszerzenie. Dokonano porównania z innymi, znanymi logikami programów.

Książka jest przeznaczona dla programistów, projektantów systemów przetwarzania informacji, pracowników nauki zajmujących się informatyką oraz dla studentów kierunków informatycznych i matematycznych.

Harold Abelson, Andrea A. diSessa: Geometria żółwia – komputer jako środek do poszukiwań matematycznych. WNT, Warszawa 1992, wyd. 1, s. 460. ISBN 83-204-1482-2

Książka jest przeznaczona dla osób interesujących się lub zajmujących się nauczaniem informatyki i zasad posługiwania się komputerem, grafiką komputerową oraz zastosowaniami informatyki w matematyce, fizyce i innych naukach ścisłych.

Bogdan Frelek: Commodore 64. WNT, Warszawa 1993 r., wyd. 4, s. 208. ISBN 83-204-1516-0

W książce podano podstawowe informacje dotyczące mikrokomputera Commodore 64 (C 64). Omówiono jego budowę, wykorzystanie oraz współpracę z urządzeniami zewnętrznymi. Opisano jego programowanie w Basicu i asemblerze. Przedstawiono możliwości uzyskiwania efektów dźwiękowych i graficznych. Podano również wiadomości o mikrokomputerach pokrewnych takich, jak: VC 20, C 16, Plus 4, C 128.

Książka jest przeznaczona dla szerokiego kręgu czytelników interesujących się użytkowaniem i programowaniem popularnych mikrokomputerów osobistych.

Dorothy Elizabeth, Robling Denning: Kryptografia i ochrona danych. WNT, Warszawa 1992 r., wyd. 1, s. 430. ISBN 83-204-1480-6

W książce omówiono zasady ochrony procesów przetwarzania danych, zagrożenia bezpieczeństwa przetwarzania danych i możliwości przeciwdziałania im. Podano podstawowe pojęcia z dziedziny kryptografii, ochrony danych, teorii szyfrowania. Omówiono sterowanie przepływu informacji i mechanizmy sterowania zapobiegające ujawnianiu poufnych danych.

Książka jest przeznaczona dla osób zajmujących się informatyką oraz dla studentów kierunków informatycznych.

EDI (*Electronic Data Interchange* = Elektroniczna Wymiana Danych) polega na przesyłaniu komunikatów elektronicznych, stanowiących zdematerializowane odpowiedniki tradycyjnych dokumentów papierowych, między systemami komputerowymi partnerów handlowych przy wykorzystaniu sieci telekomunikacyjnych. W warunkach EDI następuje zautomatyzowanie rutynowych czynności działalności gospodarczej, a w konsekwencji, znaczne potanienie i przyspieszenie obsługi transakcji (całodobowa praca ciągła przez 7 dni w tygodniu). Stwarza to praktyczne możliwości zastosowania nowoczesnych form organizacji w produkcji (JUST-IN-TIME), w handlu oraz w bankowości (np. likwidowanie tzw. zatorów płatniczych).

Gwałtowny rozwój informatyki i telekomunikacji na świecie stworzył podstawy szybkiego rozpowszechniania się EDI w gospodarce, technice i administracji. Pięćset największych światowych koncernów (wg czasopisma „Fortune") już stosuje EDI. Wkrótce zastosowanie elektronicznej wymiany danych w miejsce tradycyjnej wymiany dokumentów papierowych stanie się niezbędnym warunkiem współpracy z partnerami z krajów gospodarczo rozwiniętych. Powstanie swego rodzaju bariera technologiczna w sposobie prowadzenia działalności gospodarczej, powodująca stopniowe eliminowanie z rynku firm używających metod tradycyjnych.

W związku z aktualnością i ważnością problemu EDI oraz potrzebą rozszerzenia wiedzy, wymiany informacji i doświadczeń, a wreszcie integracji środowiska polskich użytkowników – powstała idea organizowania, podobnie jak w wielu innych krajach, dorocznych krajowych konferencji EDI. **I Krajowa Konferencja EDI odbędzie się w dniach 15–17 czerwca br. w ośrodku konferencyjnym Uniwersytetu Łódzkiego w lesie Łagiewnickim pod Łodzią** (dojazd autobusem miejskim z dworca Łódź-Fabryczna).

W Konferencji weźmie udział kilku znanych ekspertów zagranicznych oraz liczni krajowi, którzy zaprezentują referaty m.in. na następujące tematy:

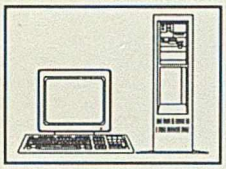
- metodologia projektowania systemów EDI,
- problemy bezpieczeństwa transakcji w warunkach stosowania EDI,
- problemy infrastruktury telekomunikacyjnej na potrzeby EDI,
- problemy prawne związane ze stosowaniem EDI,
- problemy strategii marketingowej w warunkach EDI,
- realne dokonania i doświadczenia w zakresie EDI w Polsce.

Oprócz referatów przewidziano dyskusje panelowe oraz demonstracje systemu EDI w praktycznym działaniu. Zostaną zaprezentowane osiągnięcia w tej dziedzinie trzech czołowych światowych koncernów: IBM, Digital, ICL.

Konferencja jest adresowana do wszystkich obecnych i przyszłych użytkowników EDI, a zwłaszcza do przedsiębiorstw przemysłowych, bankowych i handlowych, zaangażowanych w operacje zagraniczne. Informujemy, że **mimo upływu oficjalnego terminu zgłoszeń istnieją jeszcze możliwości uczestniczenia w tej nowatorskiej Konferencji. Szczegółowych informacji w tej sprawie udziela prof. dr hab. Marian Niedźwiedziński, ul. Szklana 2/14 m. 3, 91-844 Łódź, tel./fax: (0-42) 55-67-57**

Weiss Z.: Rozproszone systemy operacyjne INFORMATYKA 1993, nr 6, s. 1 Przystępne omówienie zasad działania oraz specyfiki rozwiązań systemów operacyjnych dla sieci komputerowych.	**Weiss Z.: Distributed operating systems** INFORMATYKA 1993, No. 6, p. 1 Popular presentation of operation principles and specific solutions of operating systems for computer networks.	**Weiss Z.: Verteilte Betriebssysteme** INFORMATYKA 1993, Nr. 6, S. 1 Eine populäre Vorstellung von Betriebsregeln und spezifischen Lösungen der Betriebssysteme für Computernetze.
Wrycza S., Plata-Przechlewski T.: Badania ankietowe. Ocena poziomu zastosowań informatyki dla celów zarządzania w Polsce INFORMATYKA 1993, nr 6, s. 6 Metodyka badań oraz analiza wyników pierwszej części ankiety na temat oceny poziomu zastosowań informatyki do celów zarządzania w Polsce.	**Wrycza S., Plata-Przechlewski T.: Inquiry research. Level's evaluation of informatics applications for management purposes in Poland** INFORMATYKA 1993, No. 6, p. 6 Research methodics and analysis of results from the first part of the inquiry on level's evaluation of informatics applications for management purposes in Poland.	**Wrycza S., Plata-Przechlewski T.: Umfrageforschung. Eine Bewertung von Niveau der Informatikanwendungen für Verwaltungszwecke in Polen** INFORMATYKA 1993, Nr. 6, S. 6 Forschungsmethodik und Analyse von Ergebnissen des ersten Teiles einer Umfrage über Niveaubewertung von Informatikanwendungen für Verwaltungszwecke in Polen.
Laurentowski A., Zieliński K.: SR – obiektowo zorientowany język programowania rozproszonego INFORMATYKA 1993, nr 6, s. 15 Charakterystyka języka SR do programowania równoległego i rozproszonego w lokalnych sieciach komputerowych.	**Laurentowski A., Zieliński K.: SR – an object oriented language for distributed programming** INFORMATYKA 1993, No. 6, p. 15 Characteristics of the SR language for parallel and distributed programming in LAN computer networks.	**Laurentowski A., Zieliński K.: SR – eine objektorientierte Sprache für verteilte Programmierung** INFORMATYKA 1993, Nr. 6, S. 15 Eine Charakteristik von der SR-Sprache für parallele und verteilte Programmierung in LAN-Computeronetzen.
Witkowski T.: Algorytm rozpoznawania problemu szeregowania zadań INFORMATYKA 1993, nr 6, s. 20 Omówienie algorytmu szeregowania zadań w rozpoznawaniu obrazów, zastosowanego do opracowania optymalnych harmonogramów pracy maszyn.	**Witkowski T.: The recognition algorithm for task arrangement problem** INFORMATYKA 1993, No. 6, p. 20 Discussion of task arrangement algorithm in image recognition, applied for elaboration of optimal machine work schedules.	**Witkowski T.: Ein Erkennungsalgorithmus für das Problem des Augabenordnens** INFORMATYKA 1993, Nr. 6, S. 20 Eine Besprechung von Algorithmus für Aufgabenordnen in Bildererkennung, der zur Erarbeitung von optimalen Maschinenarbeitzeitplänen angewendet wurde.

JUNISOFTEX®



Przedsiębiorstwo Produkcyjno-Usługowe Innowacji Technicznych Biuro Techniczno-Handlowe 44-100 Gliwice ul. Konstytucj 11.
tel. 31-57-00, 31-51-31, 31-75-10, fax 31-75-10, centr. 31-90-81 do 88, telex 036233
Zespół wdrożeń i analiz systemów komputerowych wewn. 15 ,25, 282, Zespół inżynierii oprogramowania wewn. 14, 20, 245,
Zespół d/s serwisu systemów Novell i sprzętu komputerowego wewn. 16, 245, Dział Handlowy wewn. 18, 28, 201,
Szkoła Komputerowa wewn. 282, Sekretariat wewn. 250, Prezes wewn. 13, 272, Dyrektor wewn. 12, 272, Główny Księgowy wewn. 256,
** Konto: PKO I/O Gliwice 27661-225298-136 **

OFERUJEMY

Kompleksową komputeryzację przedsiębiorstw obejmująca dostawę i wdrożenie zintegrowanych systemów komputerowych takich jak system:

finansowo-kosztowy
(automatyczne rozliczanie kosztów)
+
gospodarka materiałowa
+
środki trwałe
+
wyposażenie
+
obrót towarowy fakturowanie sprzedaży
+
kadry-płace
+
techniczne przygotowanie produkcji

Sprzęt komputerowy w tym:

Komputery
Monitory
Dyski twarde
Drukarki
(mozaikowe i laserowe)
osprzęt sieciowy
(karty sieciowe, kable zwykłe i światłowodowe)

Systemy i oprogramowanie licencjonowane

DOS
Novell
Btrieve
SQL Base
Windows

Szkolenia komputerowe

podstawy obsługi komputerów
arkusze kalkulacyjne (Qpro, Lotus, EXCEL)
edytory tekstów (WordPerfect, TAG, ChiWriter)
bazy danych (dbase IV, Clipper, Paradox)
systemy operacyjne

SOFTWARE SUPPORT FOR DOS, NOVELL 2.2, 3.11, 4.0, SQL BASE firmy GUPTA

*

OPROGRAMOWANIE DLA PRZEDSIĘBIORSTW

oferujemy własnej produkcji zakładowy system informatyczny PHU-Perfect(N) obejmujący swoim zakresem:
system finansowo-kosztowy umożliwiający automatyczne rozliczanie kosztów, atomatyczną dekretację, analizę rozrachunków
system obrotu materiałami, wyrobami, towarami (kody paskowe) w tym: stany magazynowe, ewidencje zamówień, fakturowanie sprzedaży (VAT) z automatyczną dekretacją rozdzielników kosztów i sprzedaży do systemu finansowo-kosztowego,
system ewidencji majątku trwałego i wyposażenia
system kadrowo-płacowy (SQL) z automatyczną dekretacją rozdzielników płacowych do systemu finansowo-kosztowego
system technicznego przygotowania produkcji (uk. 1993 r)
(oprogramowanie w systemie SQL BASE w przygotowaniu)

*

OPROGRAMOWANIE DLA URZĘDÓW MIAST I GMIN

oferujemy własnej produkcji system finansowo-podatkowy UMG-Perfect(N) obejmujący swoim zakresem:
system finansowo-księgowy,
systemy podatkowe obejmujące: ustalanie wymiarów, naliczanie i kontrolę płatności należności podatkowych dla dowolnych podatników i/lub płatników podatków na ich kontach i kontach dochodów ze swobodną korektą dowolnych informacji w dowolnym okresie a w tym:
ewidencja zapłat z wykorzystaniem kodów paskowych, emisja wezwań do zapłaty, naliczanie odsetek.
system ewidencji majątku

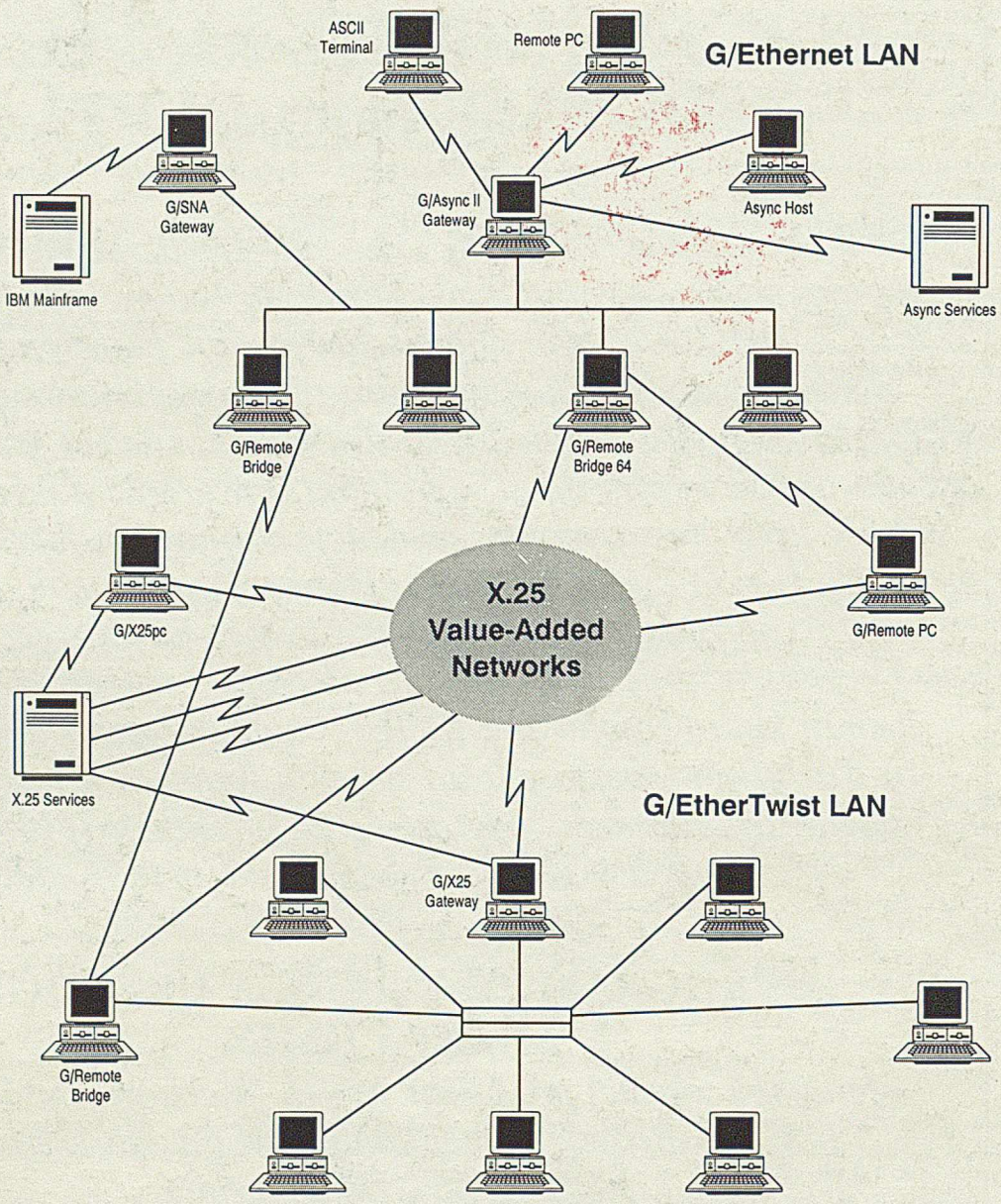
*

Zapraszamy do współpracy w zakresie wdrożeń naszych systemów partnerów, - firmy informatyczne z całego kraju !!!

0/7/92

Produkty sieciowe LAN i WAN

Gateway communications, inc.



BEZKONKURENCYJNE PRODUKTY
SIECI KOMPUTEROWYCH NAGRADZANE PRZEZ:

PC MAGAZINE, LAN MAGAZINE, INFO WORLD

OFERUJE AUTORYZOWANY DYSTRYBUTOR:

MIKROB

PC MAGAZINE EDITORS' CHOICE January 31, 1989 G/Ethernet AT

PC MAGAZINE EDITORS' CHOICE October 16, 1990 G/EtherTwist

P.W.P.T. MIKROB SP. Z O.O.

20-346 Lublin, ul. Długa 5

Tel. (0-81) 420-61, faks 415-43, teleks 643776 MIKRO PL

0/15/91

Na życzenie wysyłamy katalog produktów. Atrakcyjny program współpracy dla dealer'ów.